CERTIFICATE

This is to Certify that Group 5 with Members:

Aveepsa Das, Gitisha Mishra, Shreelu Santosh and Kunal Jain,

of

Cyber Security and Digital Forensics Division

under School of Computer Science Engineering (SCSE)

has successfully completed

Project Exhibition 1 (DSN 2098)

during Fall Semester 2022-23

under the guidance of Dr S Rajasoundaran

and have submitted a satisfactory report, as compiled in the following pages.

**Zero-Trust Network Architecture (ZTNA) based Security Policy for Vehicular Ad Hoc Network (VANET)**

Aveepsa Das, Gitisha Mishra, Shreelu Santosh, Kunal Jain

School of Computing Science and Engineering (Branch: Cyber Security and Digital Forensics)

VIT Bhopal University, India

DSN2098 (Project Exhibition-1)

Fall Semester 2022-23

Faculty Guide: Dr. S.Rajasoundran

**ABSTRACT**

These days, VANETs (Vehicular Ad hoc Networks) are increasingly becoming common. Some cities have already adopted this network for ease in managing traffic and averting road accidents.

However, like every other network used in our daily lives, VANETs are not safe from various cyber-attacks. Some of these cyber-attacks can wreak havoc on road traffic, which can lead to losses in lives and infrastructure each year.

Our project is to study Zero Trust Network Architecture (ZTNA) and implement it in the VANET network. Also, we aim to investigate the attack mechanism and detection algorithms of some of the active attacks on VANETS, namely, DDoS, man-in-the-middle attacks, blackhole attacks, and misrouting attacks. Finally, we attempt to simulate these attacks using OmNet++ and compare the impact of active attacks on packet delivery in the network.

**OBJECTIVE**

To successfully simulate a Vehicular Ad hoc Network by implementing Zero Trust Network Architecture (ZTNA)

**INTRODUCTION**

* VANET (Vehicular Ad hoc Network) refers to a network where different moving vehicles and other connecting devices come in contact over a wireless medium and exchange useful information with one another.
* A special case of MANET, in which vehicles act like moving nodes.
* Nowadays, many brands, such as Honda, BMW, and Mercedes, are using VANET in their vehicles.

The applications covered by VANETs include:

1. **Safety Applications**: Post Crash Notifications, Lane Change Assistance, BlindSpot Warning, Left-Right turn assistance, Traffic Signal Violation Warning
2. **Non-Safety Applications**: Traffic Efficiency, Comfort & Entertainment, Parking Informations, Services advertisement, Road Sensing

A VANET network consists of the following components:

1. **On Board Unit (OBUs)**

* An On Board Unit (OBU) is an embedded hardware device, which is mounted inside vehicles.
* The OBU has a read-write memory to store and retrieve information, a processor, and computational capabilities.
* An OBU can be considered as a node in a vehicle that will be mobile.

1. **Road Side Unit (RSUs)**

* The RSU are stationary units, installed at fixed locations such as on towers alongside roads and other important locations, such as turns & parking lots.
* In addition to the wireless interface, it also has a wired connection, which is used to connect it with other RSUs.

The nodes of a VANET network are connected in the following ways:

1. **V2V communication**

* Wireless communication among vehicles is called Vehicle-to-Vehicle(V2V) communication. Communication between 2 OBUs, is purely Ad hoc and peer-to-peer communication as it does not require any infrastructure support.
* In V2V, vehicles directly send and receive messages to their neighboring vehicles without any interaction by the RSUs.

1. **V2I communication**

* The communication between the vehicles and roadside infrastructure is called Vehicle-to-Infrastructure (V2I) communication or Infrastructure-to-Vehicle (I2V) communication depending on the flow of information.
* The roadside infrastructures and RSUs play a major role in V2I communication.
* This communication is mainly used to connect with the Internet and get information from a larger area.

1. **V2X communication**

* Combine the use of both V2V & V2I communication.
* This is used for extending the coverage area and forwarding information about RSU from one vehicle to another.

**Zero Trust Network Architecture**

* Zero Trust is a security model, a set of system design principles, and a coordinated cybersecurity and system management strategy based on an acknowledgment that threats exist both inside and outside traditional network boundaries.
* It is grounded on three core principles: 1. Verify every user; 2. Validate every device; 3. Intelligently limit access.
* A zero trust architecture enforces access policies based on context—including the user's role and location, their device, and the data they are requesting—to block inappropriate access and lateral movement throughout an environment.

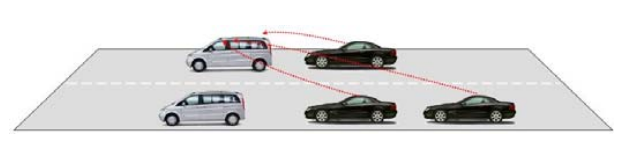
**LITERATURE REVIEW**

**Active Attacks**

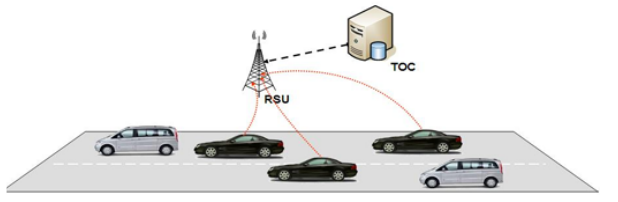
**A] DDoS Attack in VANET**

* DDOS attacks are extremely severe in the vehicular environment because the mechanism of the attack is in a distributed manner.
* The impact is dispersed in the network.
* In this kind of attack, the attackers launch attacks from different locations. We can have two possible cases for this.
* **Case 1:** Attacks are launched from different locations and each may use different time slots for sending the messages. The nature of the messages and time slots may vary from node to node of the attackers.

The attack aims to achieve network unavailability by bringing the network down at a target node. As depicted in the figure given below, there are three attackers’ nodes (black color cars) that send some messages to a target node in front (grey color car). After some time, the target node cannot communicate with any other nodes in the network.

****

* **Case 2:** In this case, the target of the attack is the VANET infrastructure (RSU) as shown in the figure below. There are three attackers in the network and they launch attacks on the infrastructure from different locations. When other nodes in the network want to access the network, the infrastructure is overloaded, thus denial of service.



**B] MITM Attack in VANET**

* MITM attacks, which pose a lot of risk in VANET, happen when a malicious node intercepts or tampers with messages exchanged between legitimate nodes.
* These malicious nodes can alter, drop or delay useful information in the network.
* In VANET, MITM attacks can be launched in 2 modes- active and passive.
* Actively, the attacker can drop, delay or change the content of received information in the network. For example, the attacker receives sensitive information such as a message about a traffic accident. The attacker changes the content of the message, or delays or drops the message. This can have a severe impact on the network as the legitimate nodes will either receive compromised messages, the information is delivered with high delays, or vehicles are prohibited from legitimate information in extreme cases.
* Passively, the attacker eavesdrops on the communication channel silently between legitimate vehicles. For instance, the MITM attacker intercepts the communication channel of law-enforcement vehicles and shares the communication with the interested organization for their own benefit.

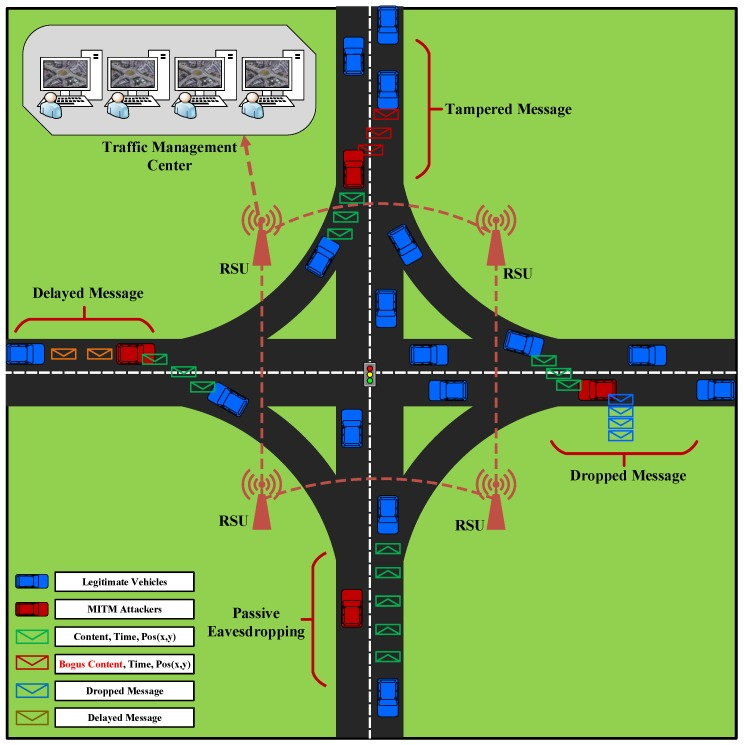
***Mathematical Model*:** When an event occurs in VANET, the transmitted packet (MG) from the message generated by the vehicle contains the following three important information- data regarding the event, location of the event generation, and time of the event occurrence.

**MG = { data , location , time }**

where the data contains the information generated by the vehicle which can be either related to safety or non-safety application. The location contains the coordinates of the event generation vehicle in terms of *x*, *y*, *and z*, which can be achieved via the GPS of the vehicle. Further, time represents the message generation time (tsend). Therefore, the above equation can be further expanded as:

**MG = { safety data , non safety data , locx , locy , locz , tsend }**

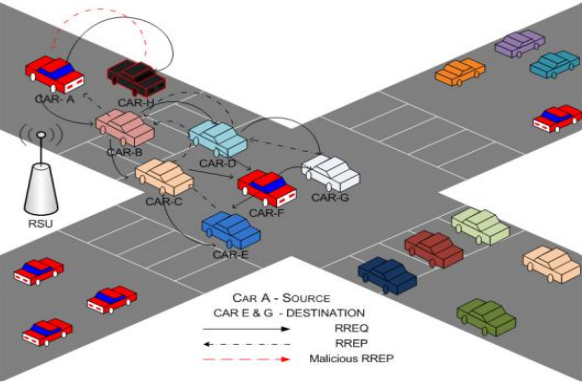
In presence of malicious nodes in the network performing a MITM attack, the attacker can compromise the content of the first equation in terms of data, location, or time. Compromising the data containing safety information can create disaster in the network.



**MITM Attack in VANET**

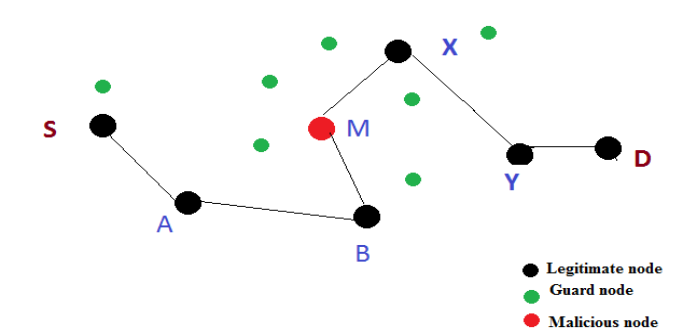
**C] Blackhole Attack in VANET**

* A black hole attack is a type of active attack that exploits the route reply message (RREP) feature of the routing protocol.
* A malicious node sends RREP messages without checking its routing table for a fresh route to a destination.
* Here, a malicious node uses its routing protocol to advertise itself as having the shortest path to the destination node or to the packet it wants to intercept.
* This hostile node advertises its availability of fresh routes irrespective of checking its routing table.
* In this way, the attacker node will always have the availability in replying to the route request and thus intercept the data packet and retain it.
* In protocol based on flooding, the malicious node reply will be received by the requesting node before the reception of the reply from the actual node; hence a malicious and forged route is created.
* When this route is established, it’s up to the node whether to drop all the packets or forward them to the unknown address.
* In the figure given below, Car-A wants to send data packets to Car-E and Car-G but it doesn’t have any route details for both. Therefore, Car-A initiates the route discovery process and RREQ is forwarded to Car-B and Car-H. As a malicious node, Car-H will claim that it is having the shortest route to reach Car-E and Car-G. Based on the available reply, Car-A will send all messages to Car-H and become the victim of a blackhole attack.



**D] Misrouting Attack in VANET**

* **In the misrouting attack, a non-legitimate node sends data packets to the wrong destination.**
* **This type of attack is carried out by modifying the final destination address of the data packet or by forwarding a data packet to the wrong next hop in the route to the destination.**
* We take the example shown in the figure below. S is sending a data packet to destination D through a route that includes <A, B, M, X, Y>.
* The malicious node M cannot misroute the data packet received from B to another node other than the next hop which is X as each guard of node M over the link B → M has an entry in its VT (verification table) which indicates X as the correct next hop.
* This fallout results in an additional scrutiny activity for the guard node involved in local monitoring, verifying that the data packet is forwarded to the correct next hop, as indicated by the entry in the guard node’s VT.
* Additionally, M cannot send another neighbor, say Q, by misrouting the packet to Q. The guards of Q over M→Q do not have an entry like <S, D, A, B, M , Q>.



**Detection of Active Attacks**

**A] DDoS Attack Detection**

Deployment of a net of honeypots [1]

* Here, a set of VANET nodes having the same computing resources as the real nodes, but with clear security flaws (known as a honeypot), are sent among the genuine nodes.
* They can be another physical vehicle or virtually, in a police car.
* They serve as bait for the attacker, who will launch a DDoS attack on one of the honeypots.
* During the attack, the data structure used to hold messages is filled up to the maximum limit.
* Once that happens, another memory data structure triggers an alert for DDoS to the driver and the police unit.
* The attacker is fooled into thinking that the DDoS attack is successful, when in fact, the real nodes are unaffected.

***WORKFLOW***:

1. In our VANET model, a few nodes (honeypots) can be provided with easy-to-crack security flaws, to attract the attacker, and a small memory to hold SYN data.
2. When the honeypot receives an SYN from another node, it checks whether the SYN belongs to the same node or not
3. The honeypot will merge all the SYN data into a cookie object and then send it back along with the ACK signal to the node. The memory sum variable is incremented
4. The sending node has two options:

a) send both the cookie and ACK to the honeypot

b) send only ACK signal

1. If both the ACK and the cookie are sent back, then the memory sum variable is decremented and the memory is maintained
2. If only ACK is sent back, then, the connection with the sending node will be disconnected and it will not affect the honeypot
3. If the memory sum variable is equal to memory size, then the memory is full, and the attack has been detected

**B] MitM Attack Detection**

Implementing an Intrusion Detection System (IDS) [2] in RSUs:

* Here, the IDS distributes a key from the cloud to all the nodes in the network.
* The nodes are expected to decrypt the key and perform arithmetic operations on the payload.
* Once the operations are executed, the nodes encrypt the packets and send them back to the IDS.
* If the returned packet fails to meet the criteria, or if the packet is taking too long to be sent, or is not sent back within a given window of time, then the IDS deems the sending node to be malicious.

***WORKFLOW***:

1. The IDS is installed in each node of the network.
2. The IDS retrieves a key from a cloud and encrypts it using AES or DES encryption algorithms.
3. The IDS then distributes the encrypted key among the neighboring nodes in the network.
4. On receiving the packet, the nodes decrypt the key.
5. The node then multiplies the payload by 2 or performs more secure computation techniques on the key.
6. The computed key generated, is then again encrypted and sent back to the IDS.
7. The IDS monitors the behavior of the nodes and records the time taken for the round trip of the packet from the node.
8. On receiving the returned packet, the IDS checks the key to see if it is the correct key and has been computed according to the pre-established protocol.
9. If the returned packet is incorrect or has not been received, then IDS declares the node as malicious and asks the genuine nodes to close connections with that node.
10. If the round-trip time exceeds the threshold time, the IDS also declares the node as malicious.

**C] Blackhole attack detection**

Enhanced RID-AODV [3]

* It involves creating and maintaining a dynamic blacklist of malicious nodes, for each genuine node in the VANET.
* Each non-malicious node will prevent sending and receiving messages to and from the nodes which are on its dynamic blacklist.
* In other words, they will not use the blacklisted nodes as intermediate nodes.
* A dynamic blacklist means that the non-malicious node will add or remove nodes from the blacklist based on specific criteria.

*Criteria*: Repetitive mismatch in hash value of the receiving frames from the same neighboring node.

* Each node keeps a counter which increments each time a mismatched hash value is encountered.
* Once the counter reaches a threshold value, the node is added to the blacklist of the subject node.
* If the node is moving at a relatively high speed, there can be a mismatch of hash values due to normal link failure.

So, given node speed, NodeSpeed, and threshold value when node speed is zero, C, the threshold value for a malicious node is given as:

**MalThresholdValue = NodeSpeed + C**

When a node is added to the dynamic blacklist, it is also assigned an expiry time.

Once the expiry time is over, the node is removed from the blacklist and its corresponding counter and expiry timer are reset.

**NodeExpTime = CurrentTime + BlockingPeriod**

**D] Use of Metrics** [4]

* A set of metrics, like identifying metrics, performance metrics, and time metrics are used to measure the flow of packets in a network in real-time.
* A set of authorized nodes (which can be implemented in a set of RSUs) check the metrics and then determine the level of traffic in the packet route.
* If the traffic is high, the authorized node notifies the nodes (OBUs) next in the route about the packet transmission that is to take place, as well as the likely time for the packet reception.
* The nodes will wait for a maximum of a set time to receive the packets, otherwise, it sends out an alert to the authorized node, who will help identify and locate the nodes which are misrouting the packets.

Each node is assigned a set of metrics namely,

1. Performance metrics, with reference to packet delivery ratio and packet drop ratio.

Each node is assigned a value between 0-1

1. Time metric, which is assigned according to the time taken to deliver a packet and receive acknowledgment

Calculating total time delay: it is calculated according to 4 conditions

1. Large Size Low Traffic (LLO)
2. Large Size High Traffic (LHI)
3. High Size Low Traffic (HLO)
4. High Size High Traffic (HHI)

***WORKFLOW***

1. All metrics of the nodes are calculated and this data is forwarded to an authorized node, which will act as an information node about all other nodes.
2. If node A wants to send and receive data from another node B, it will collect the traffic information from the authorized node.
3. Node A will choose the best path for transmitting data, based on the metrics and time delay involved. The best path is again sent to the authorized node, which will then notify the nodes in the path about the impending data transmission and the likely time of arrival.
4. The intermediate nodes will wait for a maximum of 5 microseconds for the packet to be received, or else it will send an alert message to the authorized node.
5. The authorized node will identify the nodes which are misrouting or trying to misroute, and block that node from the transmission

**Zero Trust Network Architecture**

Trust between vehicles can be described as “complete trust,” “general trust,” “neutrality,”

“general distrust,” and “complete distrust”, so,

D = {complete trust, general trust, neutrality, general distrust, complete distrust}.

Threshold of different permissions is Vn, then the security levels of a device can be divided into five levels:

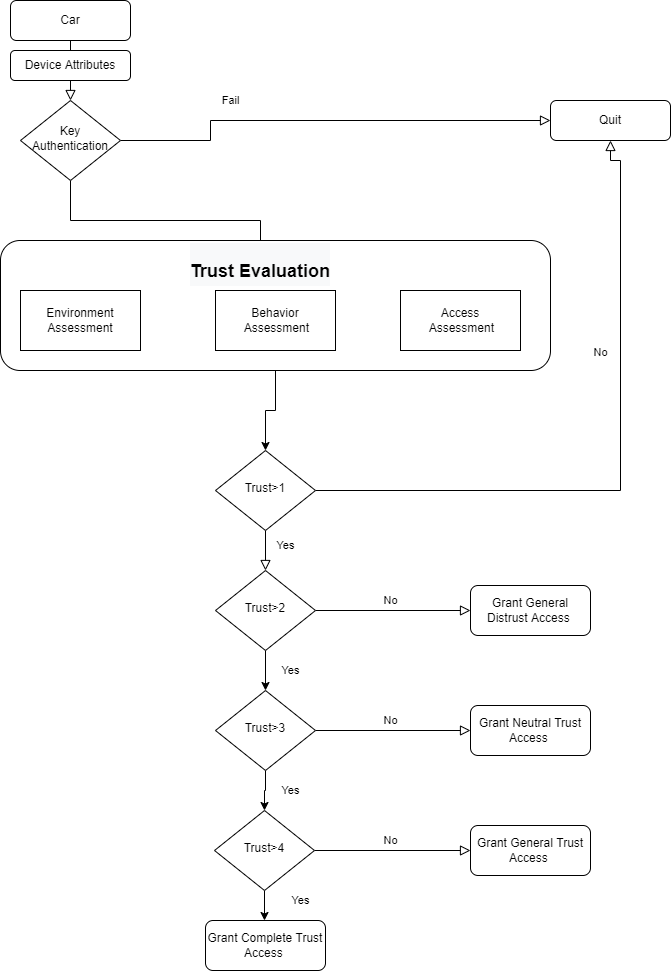
1. Complete trust (5 > X ≥ 4),
2. General trust (4 > X ≥ 3),
3. Neutrality(3 > X ≥ 2),
4. General distrust (2 > X ≥ 1), and
5. Complete distrust (1 > X ≥ 0).

Zero Trust Architecture is based on few defining parameters corresponding to which the trust value of a VANET Node is calculated,

* Key Authentication
* Access Control
* Behaviour Assessment

Based on the trust value calculated, access is provided to the node.

**Zero-Trust Architecture Workflow:**



**Key Authentication**

Stage 1 : P1 = {APi , Ei,Mi}

Where API= Authentication Packet

Ei=Encrypted Packet

Mi=MAC for node i

Stage 2 : P2={RP, EBi,MBi}

Where RP= Response Packet of base station

EBi=Encrypted Packet of base station

MBi=MAC for node i of base station

Stage 3 : P3(a) = {AKPi , EAi,MAi}

Where AKPI= Authentication Key Packet

Ei=Encrypted Authentication Key Packet

Mi=MAC for authentication key packet

P3(b)={RKP, EAi,MAi}

Where RP= Response Packet

Ei=Encrypted Authentication Key Packet

Mi=MAC for authentication key packet

P3={P3(a), P3(b)}

Stage 4 : P4(a) = {AKPi , EAi,MAi}

Where AKPI= Authentication Key Packet

Ei=Encrypted Authentication Key Packet

Mi=MAC for authentication key packet

P4(b)={RKP, EAi,MAi}

Where RP= Response Packet

Ei=Encrypted Authentication Key Packet

Mi=MAC for authentication key packet

P4={P4(a), P4(b)}

Stage 5 : P5(a) = {AKPi , EAi,MAi}

Where AKPI= Authentication Key Packet for Re-Authentication

Ei=Encrypted Authentication Key Packet for Re-Authentication

Mi=MAC for re-authentication key packet

P5(b)={ACK, RKP, EAi, MAi}

Where ACK= Acknowledgement Packet

RP= Response Packet for Re-Authentication

Ei=Encrypted Authentication Key Packet

Mi=MAC for authentication key packet

P5={P5(a), P5(b)}

**Environment Assessment**

To assess the intentions of a node, we assign an observer node to observe the activities performed by the node in the network.

T= (m+d)/(m+d+f)

where,

m = no. of packets modified

d = no. of dropped packets

f = forwarded packets without modification

T= trust assessment of node environment

An observer node is selected based on its trust level determined in the system.

**Trust Calculation**

Calculation of Initial Trust:

Tinitial= (S+U) / (Ti)

Where,

Tinitial = Initial Trust value calculation

S= Successful interaction

U= Unsuccessful interaction

Ti=Total Interactions

Calculation of Trust in Security Policy

T = A + E + R

Where,

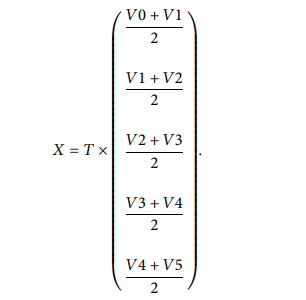
T= Trust value of the node in the security model.

A= Access control

E= Encryption of the packets

R= Secure routing of the packets

Zero Trust, X, calculation:



where,

Vn= Threshold of different permissions.

**Algorithm for Access Identification**

package org.sofwerx.ztn.access\_control;

import org.sofwerx.ztn.model.user.UserProfile;

rule "Is Group Valid"

when

$up : UserProfile( hasGroupWithDisplay( "ViewMe") )

then

$up.setValidGroup(true);

End

**Algorithm for Routing Protocol**

Initial: route request frame(RREQ), current node( currn ), previous node( pren )

While( RREQ>0 ) Do

Broadcast RREP to next node

pre currn n

currn current node

If (the energy of current node < energyTH ) Then

Drop RREQ

Break

End if

If (current node is the destination node) Then

Star the process of Routing Reply

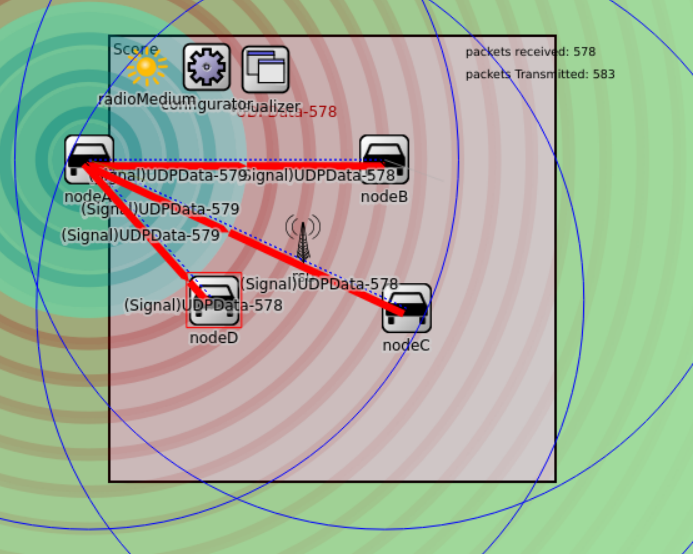
Return

End if

End While

**SIMULATION**

**Setting Up VANET network**

****

In the VANET network, we have taken:

* 1 RSU (Stationery node)
* 2 Vehicular nodes (Moving) -> nodeA & nodeB
* 2 Vehicular nodes (Stationery) -> nodeC & nodeD

**ZTNA Implementation Using Trust Values** [5]

* As discussed earlier there are 5 thresholds for trust between two vehicles, that is, *complete distrust, general distrust, neutrality, general trust*, and *complete trust*.
* Based on that, a set of categories can be set for the trust values of each VANET OBU node, namely, *‘malicious node’, ‘suspicious node’, ‘less trustworthy node’*, and *‘trusted node’*.
* Each node is assigned one of the 4 trust categories based on ZTNA trust algorithms.
* The VANET nodes having *‘malicious’* category can be reported and disconnected from the VANET network.

The nodes having *‘suspicious’* category can be monitored until their status changes to either ‘malicious’ or ‘less trustworthy/trustworthy’ categories.

The pseudocode assigning trust categories to vehicle nodes based on their ZTNA trust value is given below:

**USING ZTNA TRUST VALUES**

**Input: trust\_value (trust value assigned by ZTNA model)**

**Output: Authenticating and Connecting the Node**

**// defining trust categories**

**Define *malicious***

**trust\_value >=0 AND trust\_value <=1**

**Define *suspicious***

**trust\_value >=1 AND trust\_value <=2.5**

**Define *less\_trustworthy***

**trust\_value >=2.5 AND trust\_value <=4**

**Define *trustworthy***

**trust\_value >=4 AND trust\_value <=5**

**// determining course of action based on trust category**

**If (*malicious* ==True)**

**display.alert(“Malicious node”)**

**node\_address.report() //reports the node to monitoring personnel**

**node\_address.force\_close.connect()**

**If (*suspicious* ==True)**

**node\_address.watch() until (newcondition=*malicious* OR newcondition= *less\_trustworthy*)**

**If (*less\_trustworthy* ==True)**

**node\_address.verify() //done using digital signatures and certificates**

**node\_address.authenticate() //confirms that it is a genuine node**

**if (authenticate == True)**

**node\_address.SecureConnect() //connects to the node using strong encryption**

**If (*trustworthy* =True)**

**node\_address.authenticate()**

**if (authenticate = True)**

**node\_address.SecureConnect()**

**End**

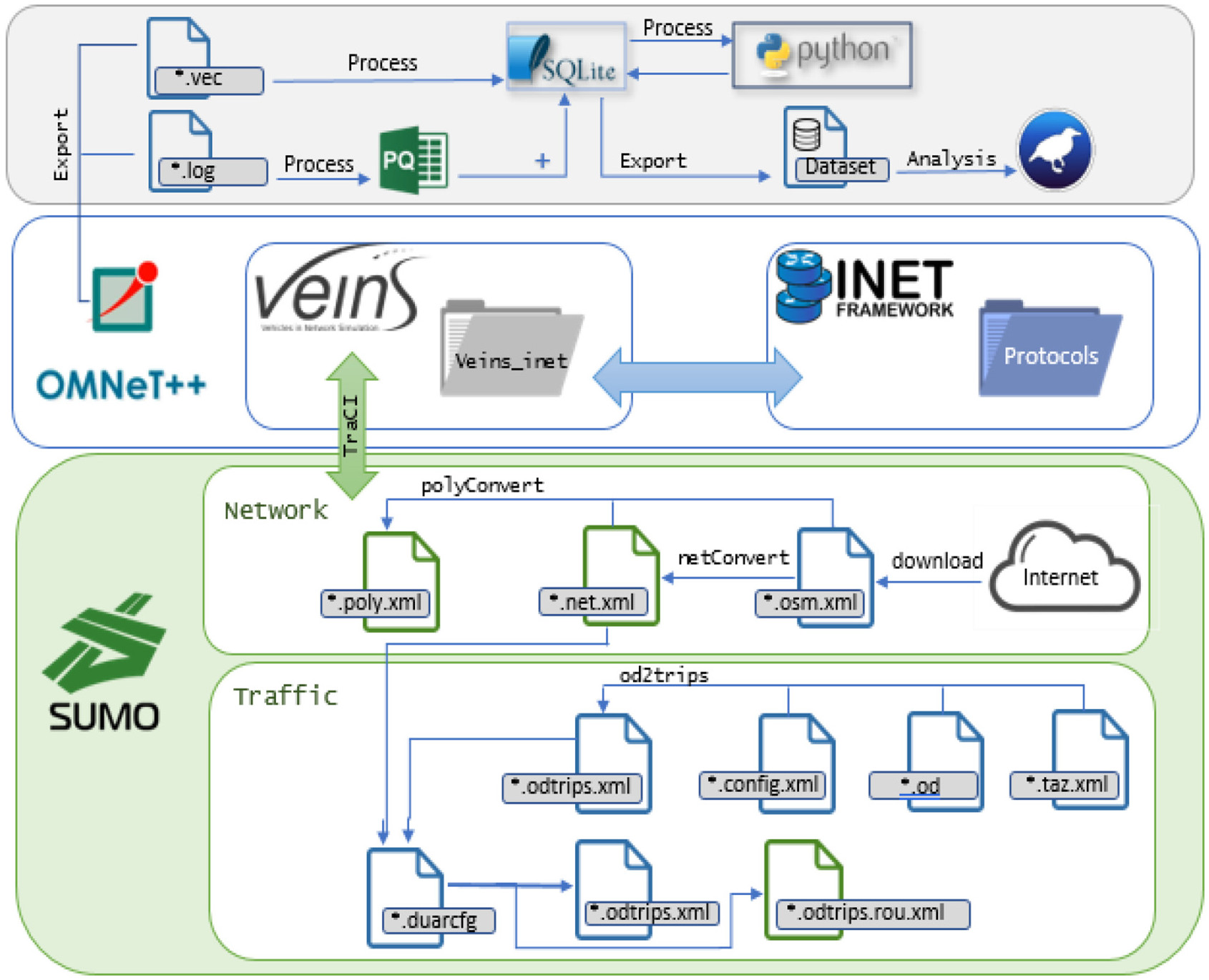
**Simulating Active Attacks on Subject Node**

**A] Algorithm to simulate DDoS Attack:**

We simulate a VANET environment considering both normal and DDoS traffic based on several simulated scenarios to be used as an input for ML methods.

The proposed work involves three main stages:

1. the first stage is to generate realistic network mobility traffic using SUMO, the second stage is to import the SUMO mobility traffic into OMNeT++ to generate the network traffic (normal and DDoS) utilising both Veins and INET and the final stage is to collect and prepare the dataset that will be used for evaluating and studying the performance of several ML algorithms.



**THE PROPOSED WORKFLOW**

* We use SUMO to prepare the network and generate the traffic and export the simulation area.
* Then, the OSM file is processed with SUMO’s net-convert utility that transforms geo-coordinates to metric coordinates of the OSM map; these metric coordinates are utilised in the next step by SUMO. The following command does this task: **netconvert –osm-files \*.osm -o \*.net.xml**
* Besides the network file, we have to consider the obstacles found within the scenario such as buildings and parks. OSM files provide such information in addition to other information like streets, lanes, junctions, and the maximum speed for each street. We use the poly-convert utility to generate a poly file, which can be used in Veins to identify all the obstacles using the following command: **polyconvert –net-file \*.net.xml –osm-files \*.osm –type-\*.xml -o \*.poly.xml**
* After generating the obstacles file, the SUMO network is established and we proceed to generate the network traffic. There are two options to generate traffic for the vehicles in SUMO.
* To simulate traffic in our network, we generate four files as follows:

1. Traffic Analysis Zone (TAZ) file which contains the edges for our route.

2. Origin/Destination (OD) matrix file that includes the origin point, the destination point, and the number of vehicles passed while taking the route.

3. Od2trips file that takes the TAZ and OD files as an input. Before generating the fourth file, we combine these three files to generate **Od\_file.odtrips.xml**, by running a specific path command.

4. SUMO configuration file that takes both the network and the OD trip files as input and then generates the route file as an output.

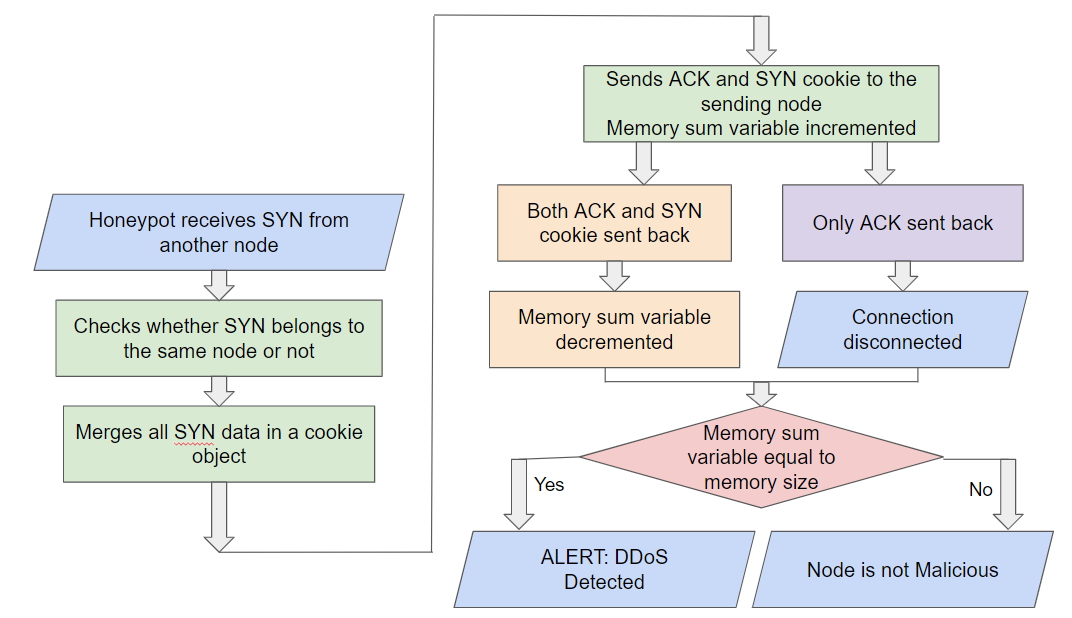
* In OMNeT++, we imported two frameworks—INET and Veins.
* We then start editing and building the simulation in three steps to meet our scenario’s parameters.
* The first step is to replace square files (square.net.xml, square.poly.xml, and square.rou.xml) with our SUMO files previously generated by the simulation and that have the same extensions.
* The second step is to edit the “scenario.ned” file to meet our scenario’s parameters and other required network configuration like the life cycle Controller which manages general operations such as shutdown, restart, suspend, and crash.
* Furthermore, we add the AODV routing protocol to the vehicle node “car.ned” to be connected with the network layer.
* The final step is to simulate both the normal and DDoS traffic through the usage of “omnetpp.ini.”

**B] Simulating MitM Attack**

* We divide the attackers randomly across the network where the attackers exist in fleet structure and the attacks are launched in a collaborative manner.
* For simulation, we use veins, which is an open source framework and is used widely for simulations of vehicular networks. It is composed of Road traffic simulator SUMO ( Simulation of Urban Mobility ) that provides traffic patterns for specific realistic map and Event-based network simulator OMNET++ (discrete event simulator) that provides various modules to ensure realistic network behaviour.
* We then create an arbitrary SUMO network and traffic. We set up a grid of any size with any number of roads and lanes and also decide the trips for a certain number of vehicles along with the duration of each vehicle. Once a vehicle reaches its destination, it leaves the simulation. All of this is stored in a config file and is used to run the simulation.
* The output holds the details of individual vehicle time, position and velocity at every step.
* We then launch Veins web server and start simulation in OMNET++ IDE. We get information and subscribe to events from SUMO through TraCI.
* Based on the events of SUMO, we send messages in OMNET++.

**Simulating Detection of Active Attacks**

**A] DDoS Attack Detection**



The pseudocode for detecting DDoS attacks is:

**Input: SYN messages**

**Output: alert when data structure is full**

**//SYN = SYN received from node i**

**syn\_ID = SYN.id() // finding SYN ID of the SYN signal**

**// i.mem => memory size of node i**

**// memory => memory of subject node’s data structure**

**// merging SYN in a cookie**

**Sum = 0**

**If (syn\_ID == i.id() AND i.mem != memory.size())**

**Cookie = SYN.merge()**

**i.send(cookie, ACK) //sends cookie and ACK back to node i**

**sum += 1**

**//course of action based on receiving cookie and ACK from node i**

**//ACK received, but not cookie**

**If (i.ack() == ACK AND i.received() != cookie)**

**connection.close()**

**// ACK and cookie received**

**Elseif (ack() == ACK AND receive() == cookie)**

**Sum -= 1 //done to maintain data structure**

**If (sum == memory.size) //when data structure is full**

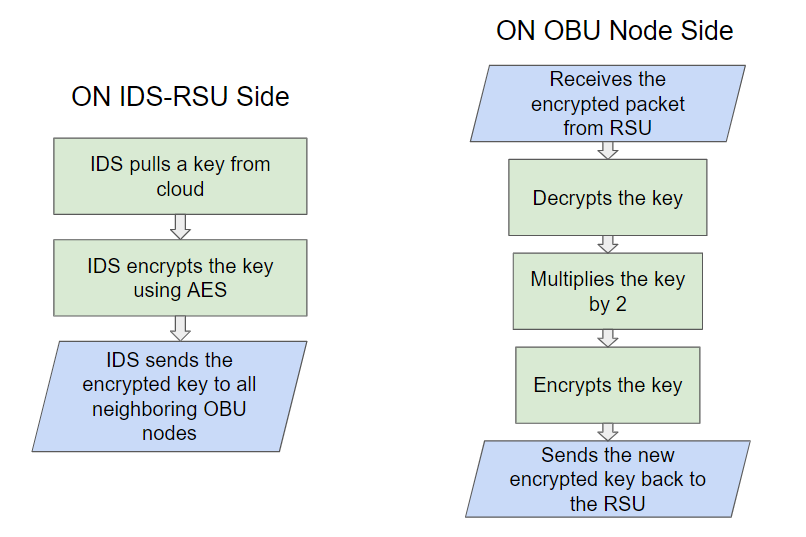
**alert(“DDOS”)**

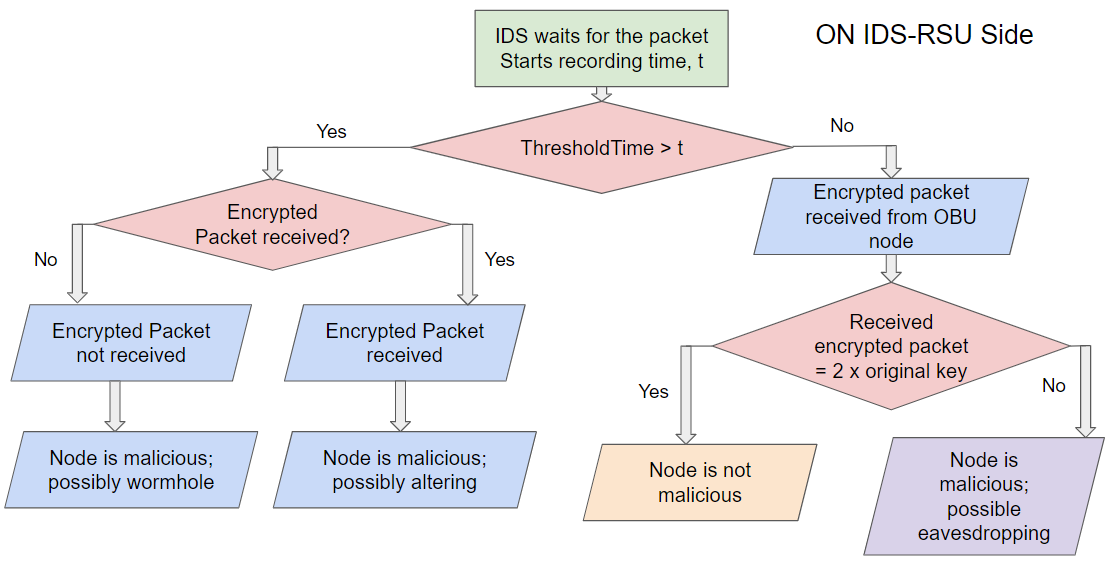
**Else**

**Do nothing**

**End**

**B] MitM Attack Detection**





The pseudocode for detecting MitM attacks is:

**Input: key, distance //network\_density= (connection\_num)/[n\*(n-1)/2]**

**Output: Checks if a node is malicious**

**//IDS RSU Side**

**A = cloud.randomkey() //key pulled from cloud**

**B = A.encrypt() // encrypting the key**

**For (node = i, i.distance <= Distance) // sending encrypted key to all neighboring nodes**

**i.send(B)**

**i.time1 = 0**

**//OBU Node Side**

**P = rsu.sent(B)**

**Q = P.decrypt() // decrypting received key**

**R = Q x 2 // multiplying key by 2**

**S = R.encrypt()**

**rsu.send(S) // sending re-encrypted back to RSU**

**//IDS-RSU Side, on receiving packet from node i**

**If (i.time2 = time() > threshold\_time) // when threshold time is exceeded**

**If (i.received() == False) // packet not received**

**alert(“Malicious Node: Wormhole”)**

**Else**

**X = i.received(S) // received packet**

**T = i.time2 - i.time1 // time taken to send and receive key from node**

**Y = X.decrypt()**

**Z = A x 2**

**// threshold time exceeded; packet received and calculated key not equal to original key**

**If (T > threshold\_time AND i.received == True AND Y != Z)**

**alert(“Malicious Node: Wormhole”)**

**// threshold time exceeded; packet received and calculated key equal to original key**

**Elseif (T > threshold\_time AND i.received == True AND Y == Z)**

**alert(“Malicious Node: Altering”)**

**// threshold time not exceeded; packet received and calculated key equal to original key**

**Elseif (T < threshold\_time AND i.received == True AND Y == Z)**

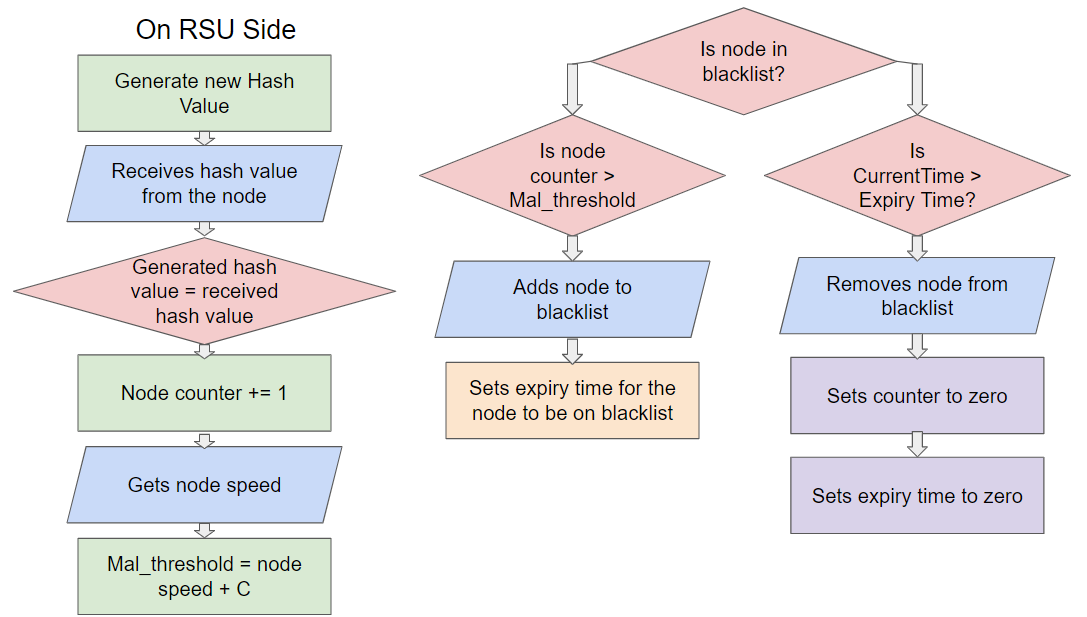
**alert(“Malicious Node: Eavesdropping”)**

**Else**

**Do nothing**

**End**

**C] Blackhole Attack Detection**



The pseudocode for detecting blackhole attacks is:

**Input: Hash Value of Node, Node Speed, node**

**Output: Adding or removing a node from dynamic blacklist**

**NewHash = hashvalue.generate() //generating new hash value**

**HashVal = node.received\_hashval() //getting hash value from node**

**If (HashVal != NewHash) //comparing hash values; not match**

**node.Counter += 1 //incrementing counter**

**NodeSpeed = node.speed() //getting moving speed of a node**

**Mal\_ThresholdValue = NodeSpeed + 5 //threshold value with C = 5**

**//adding a node to blacklist**

**//node not on blacklist and counter more than threshold value**

**If (node.isBlacklisted() == False AND node.Counter > Mal\_ThresholdValue)**

**addBlacklist(node)**

**node.BlacklistExpTime = CurrentTime + BlockingTime**

**//removing a node from blacklist**

**//node is already on blacklist and current time exceeds blacklist expiry time**

**If (node.isBlacklisted() == True AND CurrentTime > node.BlacklistExpTime)**

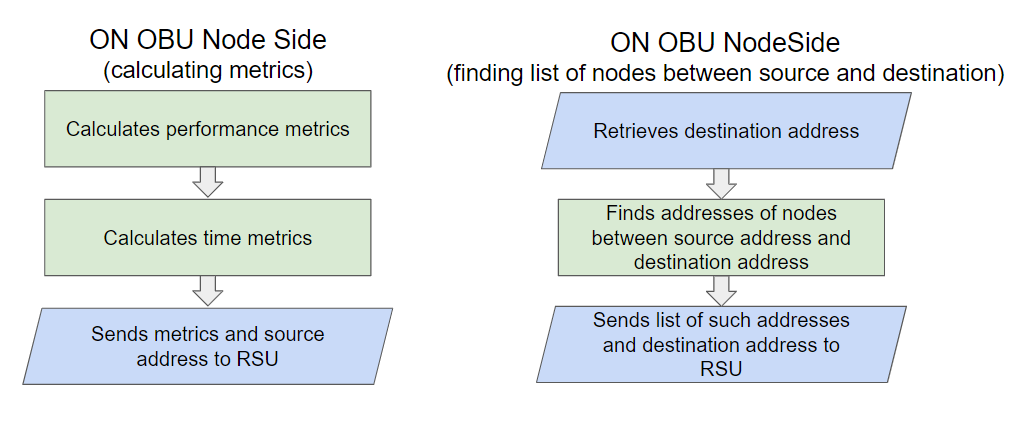
**removeBlacklist(node)**

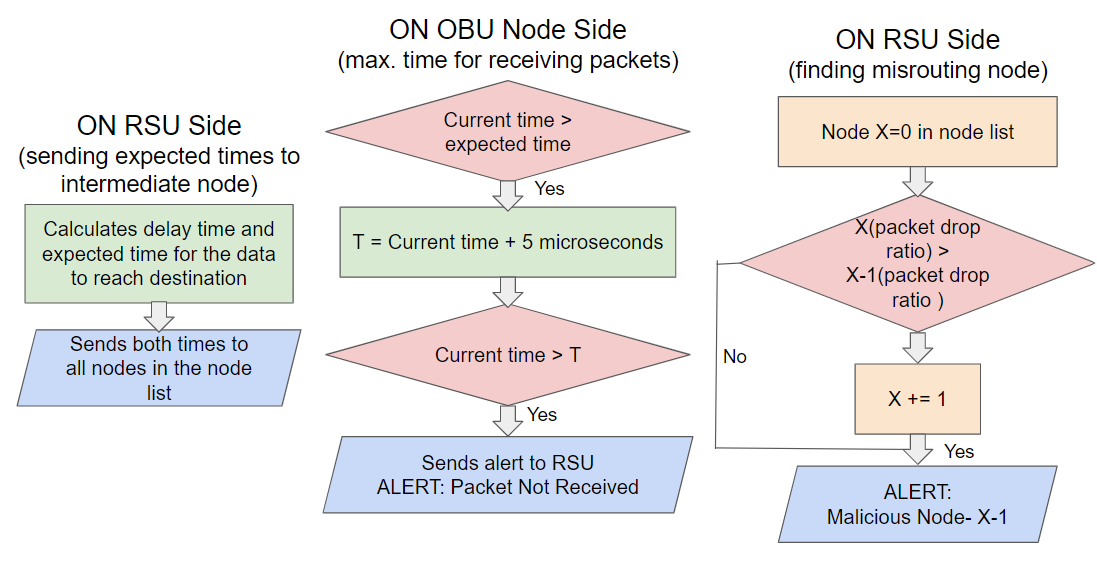
**node.counter = 0 //resetting counter and expiry time to zero**

**node.BlacklistExpTime = 0**

**End**

**D] Misrouting Attack Detection**





The pseudocode for detecting misrouting attacks is:

**Input: packets delivered, packets dropped, source and destination addresses, time delays**

**Output: Informing intermediate nodes about incoming data transmission and locating misrouting node**

**// calculating performance metrics for a node**

**Packets\_delivered = packet.delivered(time= t) //no. of packets received**

**Packets\_not received = packet.not\_delivered(time= t) //no. of packets not received**

**//adding no. of packets received and no. of packets not received**

**Sum = Packets\_delivered + Packets\_received**

**Packet\_delivery\_ratio = Packets\_delivered/Sum //packet delivery ratio**

**Packet\_drop\_ratio = Packets\_not received/Sum //packet drop ratio**

**// calculating time metrics for a node**

**//time delay in large size, low traffic conditions**

**Time.LLO = time(Traffic\_size = Medium, Low\_Traffic)**

**//time delay in large size, high traffic conditions**

**Time.LHI = time(Traffic\_size = ,Medium High\_Traffic)**

**//time delay in high size, low traffic conditions**

**Time.HLO = time(Traffic\_size = High, High\_Traffic)**

**//time delay in high size, high traffic conditions**

**Time.HHI = time(Traffic\_size = High, High\_Traffic)**

**SourceAddress = node.address() //getting address of source node**

**// list of metrics**

**Metrics = {SourceAddress, Packet\_delivery\_ratio, Packet\_drop\_ratio, Time.LLO, Time.LHI, Time.HLO, Time.HHI}**

**// sending metrics to RSU**

**rsu.send(Metrics)**

**// calculating and sending best path to RSU**

**DestinationAddress = destinationNode.address //getting address of destination node**

**If (node i in range (SourceAddress, DestinationAddress) )**

**Node\_list.append(i) //getting list of intermediate nodes**

**rsu.send(Node\_list) // sending list of intermediate nodes to RSU**

**// RSU notifying nodes in Node\_list about the incoming data transmission**

**currentTime = time()**

**//calculating time delay**

**timeDelay = delay(Time.LLO) + delay(Time.LHI) + delay(Time.HLO) + delay(Time.HHI)**

**//calculating expected time of data arrival**

**expectedTime = currentTime + deliveryTime + timeDelay**

**For (node x in Node\_list)**

**x.send(Metrics, expectedTime) //sending expected time to intermediate nodes**

**// alerting RSU about packet not received**

**If (time() > expectedTime AND packet.received == False)**

**T = time() + 5**

**If (time() > T)**

**rsu.alert(Packet\_not\_received)**

**// detecting malicious node**

**For (node y in Node\_list)**

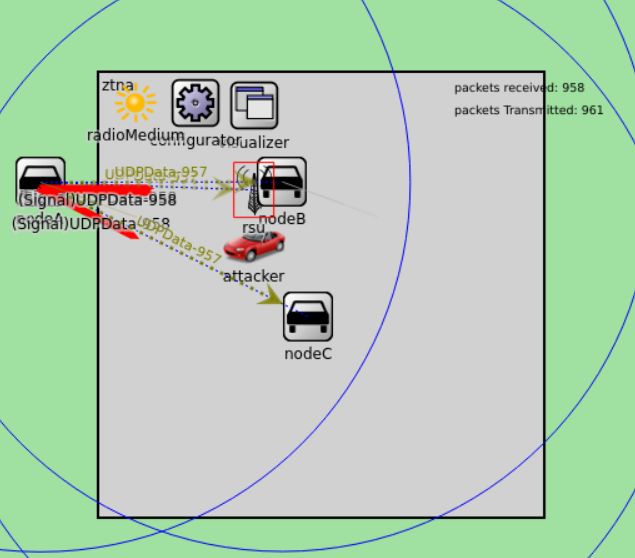
**check(Metrics) //verify metrics for change in traffic**

**// if packet drop ratio of a node is higher than the previous node**

**If (y{Packet\_drop\_ratio} > y-1{Packet\_drop\_ratio})**

**alert(“Malicious Node: Misrouting at node y-1”)**

**RESULTS**



Due to Zero Trust Policy, the ID of the nodes is acknowledged and permission is given only to trusted nodes. Any attacker or outside stranger will not be able to gain access to any information (packet) that is being transferred. Hence, malicious nodes won’t be able to perform any kind of attack in between the transfer of a packet.

**CONCLUSION**

In this project we have proposed a Zero Trust Architecture to secure the communication channels of a VANET system.

We have devised pseudocode based on trust values under Zero Trust Network Architecture and implemented it in a VANET simulation.

We have also provided flowcharts and pseudocode for the detection of four active attacks on VANET nodes, namely DDoS attack, man-in-the-middle attack, blackhole attack, and misrouting attack.

We have simulated a VANET network using the OmNet++ tool and implemented the algorithms for launching attacks on the subject node in the network.

**REFERENCES**

1] Manish Naik, “Early Detection and Prevention of DDoS Attacks in VANET”, National Institute of Technology, Rourkela, 2014

2] Farouq Aliyua , Tarek Sheltamia, Elhadi M. Shakshuki, “A Detection and Prevention Technique for Man in the Middle Attack in Fog Computing”, The 9th International Conference on Emerging Ubiquitous Systems and Pervasive Networks, 2018.

3] Rushdi A. Hamamreh, “Protocol for Multiple Black Hole Attack Avoidance in Mobile Ad Hoc Networks”, Recent Advances in Cryptography and Network Security, 31 Oct. 2018

4] K.Jayabharathi, A.V.Sindhuja, C. Priyanka, K. Chitra and Sanjay Kumar Suman, “MISROUTING ATTACK DETECTION USING TSM ALGORITHM”, International Journal of Advance Research and Innovative Ideas in Education, 2018

5] S. Ananthakumaran, M Sathishkumar, R.Bhavani, R.Ravinder Reddy, “Prevention of Routing Attacks using Trust-Based Multipath Protocol”, International Journal of Advanced Trends in Computer Science and Engineering, 2020

**CONTRIBUTIONS**

**Aveepsa Das (21BCY10002)**

Active Attacks on VANETs

**Gitisha Mishra (21BCY10054)**

Zero Trust Network Architecture and its workflow

**Shreelu Santosh (21BCY10090)**

Detection Algorithms and Pseudocode for Active Attacks

Pseudocode for implementing ZTNA using Trust Values

**Kunal Jain (21BCY10248)**

Simulations