TrustMod: a trust management module for NS-3 simulator.

HAJAR, M.S., KALUTARAGE, H. and AL-KADRI, M.O.

2021

TrustMod: A Trust Management Module For NS-3 Simulator

Muhammad Shadi Hajar
School of Computing
Robert Gordon University
Aberdeen, United Kingdom
m.hajar@rgu.ac.uk

Harsha Kalutarage
School of Computing
Robert Gordon University
Aberdeen, United Kingdom
h.kalutarage@rgu.ac.uk

M. Omar Al-Kadri
School of Computing & Digital Tech.
Birmingham City University
Birmingham, United Kingdom
omar.alkadri@bcu.ac.uk

***Abstract*—Trust management offers a further level of defense against internal attacks in ad hoc networks. Deploying an effective trust management scheme can reinforce the overall network security. Regardless of limitations, however, security researchers often use numerical simulations to prove the merits of novel methods. This is due to the lack of an adequate testbed to evaluate the proposed trust schemes. Therefore, there is a demanding need to develop a generic testbed that can be used to evaluate the trust relationship for different networks and protocols. This paper proposes TrustMod, an NS-3 module consisting of three main components to evaluate the different trust relationships: direct trust, uncertainty, and indirect trust. It is designed to meet usability, generalisability, flexibility, scalability and high-performance requirements. A series of experiments involving 1680 simulations were performed to prove the design and implementation accuracy of TrustMod. The performance results show that TrustMod's resource footprint is minimal, even for very large networks.**

***Index Terms*—Trust Management, simulator, NS-3, internal attacks, testbed, on-off attacks.**

I. Introduction

Maintaining the security of wireless infrastructure-less networks such as Mobile Ad hoc Networks (MANETs) is still challenging. In such networks, all nodes are expected to act as routers by forwarding packets for other nodes. The cooperation between these nodes is mandatory for network operation. Although authentication and encryption are essential security requirements, they may not be sufficient to cope with other nodes' free will. Authenticated nodes can still get compromised or act selfishly to disrupt the overall network operation. Therefore, deploying an effective Trust Management System (TMS) can monitor the distributed collaborations between nodes to differentiate between trustworthy and untrustworthy nodes. Moreover, several applications emerge from deploying a TMS ranging from security applications such as malicious node detection and access control to routing [1]–[3].

Abundant research is put forward to evaluate the trust relationship using different approaches and techniques [3]. However, the majority of these trust schemes are evaluated using numerical simulations due to the shortage of a dedicated simulator or a testbed to simulate the trust relationship. Trust evaluation using numerical computing environments such as MATLAB can only provide an abstract mathematical and numerical analysis for the proposed trust schemes. However, these unrealistic experiment setups can not reflect the actual network operation where tens of protocols from different stack layers work collaboratively. For instance, traffic rates, packets re-transmissions, collisions, and routing protocols are all examples of network operating conditions and protocols that can not be fulfilled using existing numerical analysis tools, including MATLAB. This paper therefore presents TrustMod, a new testbed environment for NS3 [4] to realistically simulate the trust relationship on ad hoc networks.

Among a wide range of network simulators, NS-3 [4] has been chosen to build this testbed for several reasons. It is an open-source network simulator designed essentially for research. It provides a robust core written in C++ with high compatibility and scalability characteristics. Unlike NS-2, which uses OTcl to write the simulation scenario, NS-3 uses C++ with Python bindings that allow researchers to import NS-3 libraries as Python modules. Moreover, the performance analysis of NS-3 shows an optimal trade-off between memory consumption and simulation run-time among different network simulators [5].

The main contribution of this paper is introducing a trust management module for NS-3. Our proposed NS-3 module, TrustMod, consists of three main components, which are direct trust evaluation, uncertainty evaluation and indirect trust evaluation. It has been designed to be resource efficient and easily integrated as other NS-3 modules. Furthermore, the code of our proposed trust management module is made available at (*https://github.com/mshsyr/TrustMod*).

The remainder of this paper is organized into seven sections as follows. Related work is given in section II. Section III provides an overview of. The proposed trust management module is introduced in section IV, followed by the module validation results provided in section V. The performance evaluation results is presented in section VI. Finally, section VII concludes the paper.

II. Related Work

Many different trust schemes are proposed in the literature for different wireless networks and applications. However, due to the unavailability of more realistic trust evaluation simulators or testbeds, MATLAB is still the first choice for researchers to evaluate their schemes' effectiveness. Few tools are introduced in the literature to evaluate trust schemes, such as TOSim [6]

and TRMSim-WSN [7]. In [6], authors introduced a tool to evaluate various types of trust and reputation schemes targeting overlay networks with four threat models. It is a scope-specific tool, which is not applicable to other networks. In [7], authors introduced a simple java-based trust simulator for Wireless Sensor Networks (WSNs). This simulator allows the researchers to tune several parameters, such as nodes number and delay, to simulate different kinds of malicious activities. However, as the protocol stack is not implemented in TRMSim-WSN, it is regarded as a conceptual simulator that can not reflect the realistic network's behaviour. The main shortcomings of using the aforementioned tools are the incapability of simulating the targeted network operating conditions and the inability to integrate with other stack layers; therefore, the proposed trust schemes can not be evaluated under different stack protocols, such as routing protocols.

On the other hand, TrustMod is a generic trust management testbed built as a module for NS-3. It works as a cross-layer module in the TCP/IP stack. Therefore, to the best of our knowledge, it is the first trust management testbed that is fully integrated with the protocol stack and can be used with different networks and under different network conditions and parameters. It can be added like any other built-in modules to the nodes, which are the conceptual computing devices in NS-3. Researchers can use the NS-3 attributes system to configure the required trust management properties and then write their scheme implementation.

III. Network Simulator 3

For decades, NS-2 [8] was regarded as the de-facto standard simulator for research. Countless published research papers are evaluated using NS-2. In 2006, a project to develop a new network simulator to replace NS-2 was begun. NS-3 is built from scratch using C++, although some models are ported from predecessor simulators such as NS-2, YANS [9] and GTNetS [10]. The initial release of NS-3 was available in 2008, while the development and maintenance of NS-2 stopped in 2010 [11]. The main aim was to enhance the NS-2 models' realism by making it closer to how real computers operate. For instance, NS-3 adopts the Linux architecture for sockets and internal interfaces. Moreover, NS-3 supports emulations by incorporating real network devices to form a real testbed.

NS-3 is built as software libraries that can be linked to the simulation scenario statically or dynamically. Fig. 1 describes how the NS-3 modules are organized. Module dependencies may usually exist with other underneath modules. The core module of NS-3 consists of C++ classes that provide time services, smart pointers, callbacks, debugging facilities, and other significant services. These services are used by all kinds of hardware, protocols, and environmental models. The list below contains other important NS-3 components:

- Network module, which models the network packet, packet tags and packet headers. Moreover, node class and the abstract base class netdevice are defined in this module, in addition to address types such as IPv4 and MAC.
- Mobility module, which has different mobility models such as static, random and walk.
- Helper API contains classes and methods that provide high-level wrappers to encapsulate low-level API calls. It is widely used when scripting a simulation scenario.

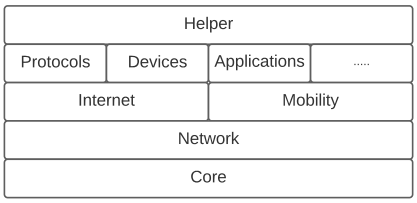


Fig. 1: NS-3 Modules Architecture

NS-3 allows researchers to evaluate their simulation models using different kinds of tools and systems. A logging facility is used to monitor and debug the execution of the simulation scenario. It can be enabled from the simulation script for different NS-3 components using the environment variable NS_LOG. Logging messages are classified into different severity classes, which can be set by the user to see the log of a specified severity class for a specified NS-3 component. The tracing system is another tool to allow users to gather information and statistics to evaluate their simulation models or modifications. NS-3 defines two independent components, tracing sources and tracing sinks, where the connection between the sources and the sinks can be established with the help of the attributes and the callback mechanisms. NS-3 provides two kinds of tracing output. The first is similar to NS-2, where all events associated with their properties, like timestamps, are output to text files. The second tracing output is PCAP binary files for capturing live traffic. PCAP files can be analyzed by TCPDUMP or Wireshark applications. FlowMonitor is an NS-3 module for monitoring traffic flows contributed by authors in [12]. It provides an easy tool to monitor the traffic across the network. It allows researchers to measure the performance of their methods by providing different kinds of statistics such as packet loss rate, delays and bit rates. Furthermore, although NS-3 does not have a built-in graphical animation tool, an offline animator toolkit can be used for visualization. NetAnim is an example of those tools. It uses the generated XML trace file for animation. Animation includes but is not limited to transmitted packets over different links, packets timeline with filtering capabilities, IP and MAC information and routing tables.

IV. The Proposed Trust Management Module

In this section, our trust management module for NS-3, termed as TrustMod, is introduced. The design requirements are first identified, and then the module structure and its components are presented.

A. Requirements

Different objectives are taken into account when designing our trust management module for NS-3. These objectives cover

various aspects such as usability, generalisability, flexibility, scalability and performance.

Usability is the first objective of our TrustMod module. Writing the simulation scripts is already a time-consuming and complicated task. Long times are spent on writing simulation scenarios and analyzing trace files. Therefore, the trust management module must be easy to use. This gives the researchers more time to spend on their proposed methods. The researchers can enable the trust management module for any node by adding a few code lines. Moreover, they can instantiate different trust instances for different nodes. The module design should also comply with other NS-3 modules; thus, the researchers can easily configure and set the modules instance attributes using the same way as other modules.

Trust management systems may generally rely on two components, direct and indirect trust evaluation. In direct trust evaluation, the trustor evaluates the trustee based on direct observations. However, when the trustor does not have sufficient observation history, it can ask for recommendations from other network entities. This approach is widely adopted in the literature [13]–[16]. However, obtaining recommendations is proved to be a time and resource-consuming process [17]. Therefore, the trust evaluation must comprise three main components. The direct trust evaluation process, which is based on direct observations. The indirect trust evaluation, which is based on received recommendations from other entities in the network, in addition to introducing uncertainty evaluation component that can manage the process of asking for recommendations with a view to preserve resources.

The trust module must work with different protocols and applications. Therefore, researchers can evaluate their methods for different networks and under different conditions. The distributed approach provides more flexibility to support other services and protocols within the node itself, such as secured routing protocols and access control mechanisms.

Obviously, deploying a trust module will consume some resources. However, memory and processing overhead must be as low as possible to significantly maintain high simulation performance when increasing nodes. Therefore, the simulation time and the memory footprints are expected to be minimal.

It is imperative that the proposed trust module can output all the results and statistics in a readable and easy-to-use format. This output data has to be stored by the end of the simulation process and retrieved later by researchers for results analysis. Several output format candidates are available such as binary files, ASCII files, XML/JSON files, or databases. Choosing the appropriate file format that allows the results to be stored and retrieved efficiently is essential. TurstMod uses ASCII format for output, which provides an effortless and readable way to obtain the simulation results. The trust results are formatted in a way that allows researchers to import them to Excel readily.

B. Design Overview

NS-3 network entities are defined using Node class, which is a conceptual model that other objects can be aggregated to it. The implementations of TCP/IPv4, TCP/IPv6 and other related protocols are available in the internet stack module, which can be installed into each node using the helper class. Sending and receiving packets using this module goes through different layers from NetDevice to Application classes. Trust management aims to monitor the behavior of others. Therefore, cross-layer information should be received from different sources. This cross-layer architecture allows TrustMod to receive all the needed information from other stack protocol. The received information is then processed and stored in a dynamic data structure, which will be presented in the next section. Fig. 2 provides an overview of the TrustMod structure showing the implementation of receiving the cross-layer information from both sending and receiving paths.

The TrustHelper class is used to initialize the trust module and aggregate it to the node. Two cross-layer information sources from layer 2 and layer 3 are used to evaluate the trust relationship. The first is NetDevice from the packets sending path. The SentInput method is called, and the transmitted packet and the destination MAC address is passed to it. In order to receive cross-layer information of the forwarded packets from layer 3, the promiscuous mode must be enabled. Therefore, the SetPromiscReceiveCallback has to be registered on the NetDevice in addition to writing the implementation of the method PromiscReceive. This allows layer 3 to receive the sniffed packet from lower layers, which in turn passes it with the source MAC address to the ForwardedInput method.

The observed interactions form sequences of discrete-time data. This period is set by a predefined attribute called TimeUnitAttribute, which can be set from the simulation script. This attribute timely controls all the operations inside the Trust module. Therefore, trust evaluation is scheduled at the end of each time unit by calling the method TrustSchemeManager in the TrustScheme class. Researchers have to write the implementations of their trust schemes in the following methods:

- DirectTrustEvalaution method: This method is used to evaluate the trust value based on direct observations.
- UncertaintyEvalaution method: This method is introduced to specify when to consider second-hand information from other neighbors.
- RecommendationsEvalaution method: This method is used to evaluate the received recommendations and filter out those untrustworthy ones.
- OverallTrustEvalaution: This method is called once having all the required information in order to evaluate the overall trust value.

TrustMod inherits from the Application class to allow sending and receiving recommendations. Once TrustMod is installed using TrustHelper, it initializes the recommendations listener in order to receive both recommendation requests from other neighbors and the expected recommendation responses for the recommendation requests sent by the node itself. The listening port is specified by setting the predefined attribute *RecommendationListenerPortAttribute* from the simulation script. A receive callback is defined for the receiving socket. Therefore, received packets are processed in the *RecommendationReceive*

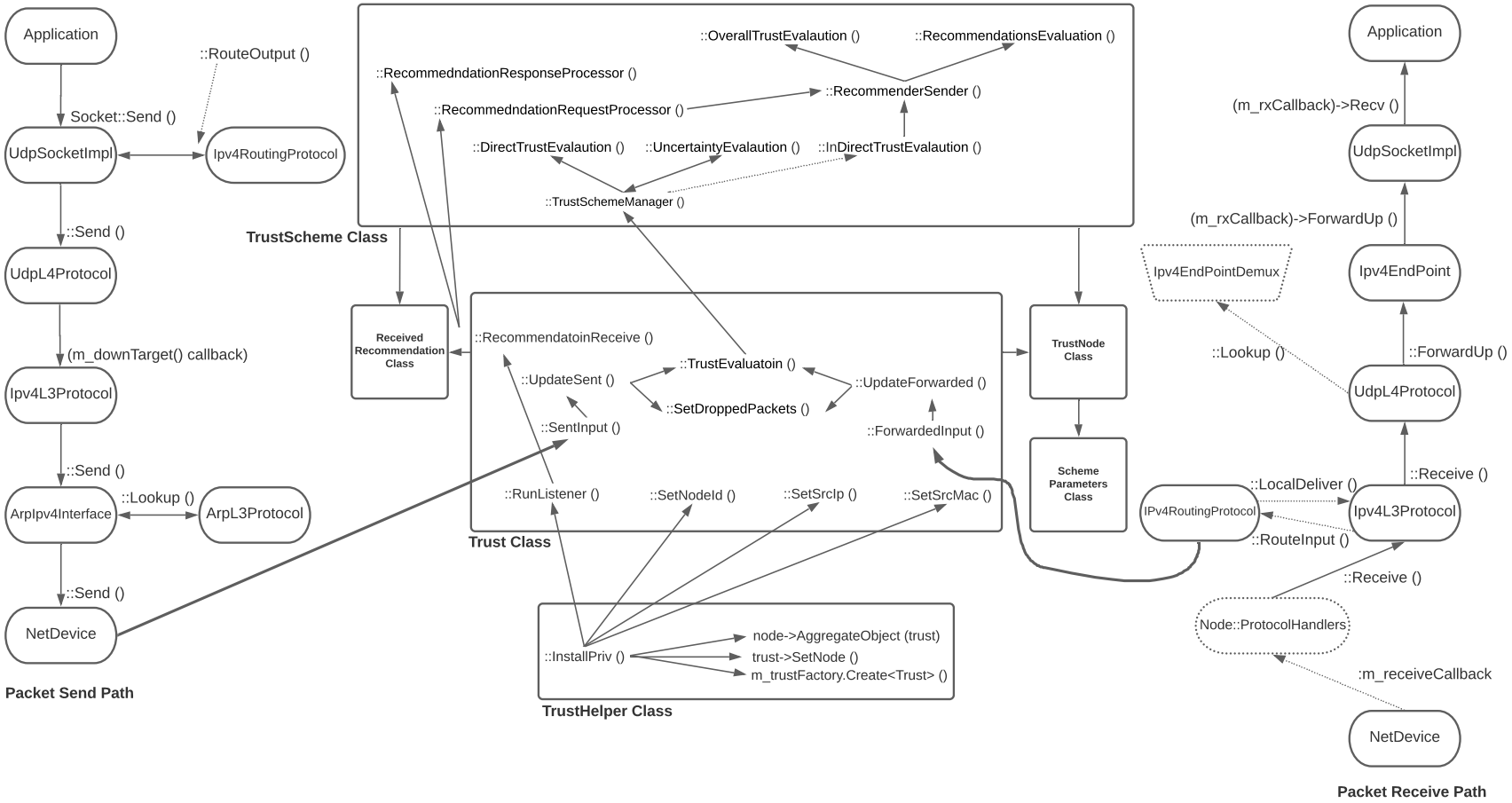


Fig. 2: Trust Module Overview

method in order to differentiate between requests and responses. Based on the received packet type, a recommendation processor is called from the *TrustScheme* class. Fig. 3 and fig. 4 show the recommendation packet format for both request and response types. Both of them have packet type, trustee MAC, trustor IP address and TrustNode (TN) sequence number. The packet type field is used to differentiate between request and response packets, while the trustee MAC field is used to specify the node in question. Trustor IP is set to the IP address of the recommendation sender. The TN sequence number represents the sequence number of the TN when the trustor is uncertain about the trustee's trustworthiness and needs second-hand information to evaluate the overall trust value. It is used to indicate to the TN object where the second-hand information is needed, and its value will be used in the recommendation response packet. On the other hand, the recommendation response has two more fields. The first contains the trust value, while the second represents how certain is the recommender about this trust value.

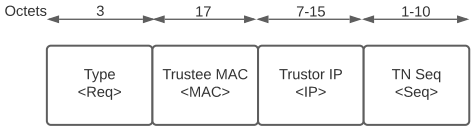


Fig. 3: Recommendations Request Format

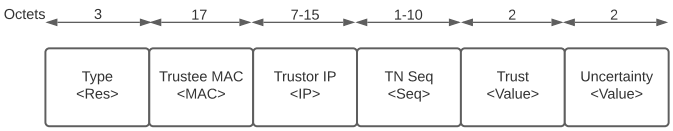


Fig. 4: Recommendations Response Format

C. Trust Module Data Structure

TrustMod receives cross-layer information and recommendation responses during the simulation process. This information is usually processed at the end of the time unit. Moreover, researchers expect detailed statistics and results at the end of the simulation. Therefore, the aforementioned information needs to be saved efficiently during the simulation. Two classes are defined for this purpose. The first is the TrustNode (TN) class to save the observations, while the second is the ReceivedRecommendations (RR) class to save the received second-hand information.

TrustMod instantiates a TN object for each time unit to save the observations and other related information. Therefore, an efficient data structure is required to store a series of TN objects for each trustee. In TrustMod, a map of vector pointers is adopted as shown in Fig. 5. The map key is set to be the trustee MAC address, while the mapped value is a pointer to the vector of TN objects. Each TN has different attributes to save observations, such as m_forwardedPackets and m_droppedPackets. In addition to the primitive attributes, there are two objects. The first is used to track the trust scheme parameters changes, while the second is used to store the received recommendations. As the recommendation request is broadcasted over the network, it is expected to receive recommendations from multiple nodes. These recommendations are stored for further processing to filter out dishonest ones. TrustMod uses a vector of ReceivedRecommendations object pointers to store the received recommendations. Adopting this data structure, as mentioned earlier, allows a dynamic memory allocation. Moreover, using pointers makes the management of data structures more efficient by saving memory and reducing the module's complexity.

D. Trust Module Implementation

The proposed trust management module is implemented using six classes. The Trust class represents the core of the module. It is responsible for initializing the trust module for each node, linking with other stack layers, listening for recommendation requests, and processing the observed information. TrustHelper class provides a separate API on top of the core NS-3 API. This helper class makes the use of the TrustMod module easier by creating and configuring the trust module effortlessly. Moreover, TrustMod has two additional classes for modeling the data structure of the module. The first is TrustNode class, which stores all the statistics and trust information for a one time unit during the simulation. It provides the necessary attributes and methods to store and evaluate the trust relationship. The second is ReceivedRecommendations class, which stores and processes the received recommendations for each sent recommendation request. On the other hand, researchers propose different methods to evaluate the trust relationship, which usually introduce new parameters and mechanisms. Therefore, two classes are used to integrate the researchers' methods into the trust module. The first is SchemeParameters class, which can be used to define scheme proprietary parameters. The second is TrustScheme, which provides the core of trust evaluation. This class is designed to have all the evaluation operations, such as processing the two kinds of recommendations request and recommendations response packets. Moreover, four methods are provided for researchers to implement their trust scheme as detailed in section IV-B. All the required information is passed as arguments to these methods with a view to providing all the necessary information for any proposed trust scheme. These arguments are passed as pointers to minimize memory and processing overhead.

All NS-3 modules are located in the src directory, where the directory's name is the name of the module. TrustMod is organized into six directories in addition to a wscript file, as illustrated in Fig. 6. The TrustHelper class source code is available in the helper directory. The model directory contains Trust, TrustNode, ReceivedRecommendations, SchemeParameters and TrustScheme classes. The bindings directory contains files related to Python bindings, while the test directory includes required module test files. Simulation examples are provided in the examples directory, while manual and useful instructions are provided in the doc directory. Finally, as all NS-3 modules

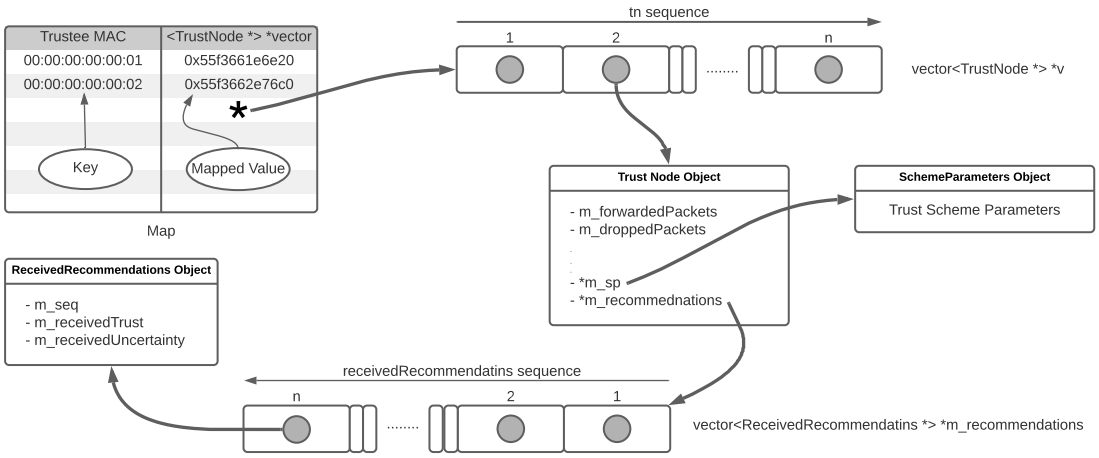


Fig. 5: Trust Module Data Structure

depend on core modules, these dependencies are defined in the wscript file. Moreover, all module source and header files must be defined there. The wscript file can be regarded as a Makefile.

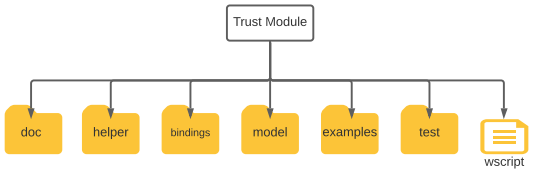


Fig. 6: TrustMod Organization

V. Module Validation

In this section, we validate the results obtained by the TrustMod module. The proposed trust scheme LTMS [18] is adopted for the validation process. LTMS is a lightweight trust management system for Wireless Medical Sensor Networks (WMSNs). It uses a novel mechanism to evaluate the direct trust value. Its results showed an outstanding performance in detecting sophisticated attacks such as on-off attacks. Simulation parameters of LTMS have been adopted to validate our trust management module by comparing the results obtained using TrustMod module with those obtained by analyzing trace files. LTMS proposes two methods to evaluate direct trust, which is regarded as the core of trust management. The first method LTMS(1), is a lightweight method used for in-body sensor nodes (SNs) as those SNs suffer from a stringent resource limitation, while LTMS(2) provides a further level of protection from on-off attacks and is designed for on- and off-body sensor nodes. Both LTMS(1) and LTMS(2) method are combined in algorithm 1, where α and β are the beta probability distribution levels, b_t and d_t are the slopes at the time unit t, $Rep_{ij}(t)$ is the reputation value maintained by the trustor i for the trustee j, $thr1$ is the defined threshold to differentiate between trustworthy and untrustworthy agents, which is usually set to 0.5 in the literature [17]–[20], $thr2$ represents the minimum trustworthiness for agents in normal operation and is set to 0.85 [18], $ShRep_{ij}(t)$ represents the short-term reputation value at the time unit t, and $cycle$ and $malicious$ are two parameters used to detect on-off attacks.

Algorithm 1: LTMS direct trust evaluation

Input: Observations & beta shape parameters
Output: Trust value
initialization;
while *true* **do**
  if $b_{t-1} \leq 0$ && $d_{t-1} > 0$ **then**
    $\alpha_t = \lambda(\alpha_{t-1} + b_{t-1}) + s(t)$;
    $\beta_t = \lambda(\beta_{t-1} + d_{t-1}) + u(t)$;
    $b_t = \alpha_t - \alpha_{t-1}$;
    $d_t = \beta_t - \beta_{t-1}$;
  else
    $\alpha_t = \lambda.\alpha_{t-1} + s(t)$;
    $\beta_t = \lambda.\beta_{t-1} + u_(t)$;
    $b_t = \alpha_t - \alpha_{t-1}$;
    $d_t = \beta_t - \beta_{t-1}$;
  end
  if $\alpha_t \leq 0$ **then**
    $Rep_{ij}(t) = 0$;
  else
    $Rep_{ij}(t) = \frac{\alpha_t}{\alpha_t + \beta_t}$;
  end
  if $Trust_{ij}(t-1) \geq thr_1$ && $Rep_{ij}(t) < thr_1$ **then**
    if $malicious > 0$ **then**
      $cycle = t - malicious$;
      $malicious = 0$;
    else
      $malicious = t$;
    end
  end
  if $cycle > 0$ && $Trust(t-1) < thr_2$ **then**
    $ShRep_{ij}(t) = mean(Trust_{ij}(t - cycle : t))$;
    $Trust_{ij}(t) = min(ShRep_{ij}(t), Rep_{ij}(t))$;
  else
    $Trust_{ij}(t) = Rep_{ij}(t)$;
    $cycle = 0$;
  end
end

The traffic is generated using the parameterized exponential density function shown in Eq. 1, which makes the number of packets different each time.

$$p(x;b) = \begin{cases} \mu e^{-\mu x} & x \geq 0 \\ 0 & x < 0 \end{cases} \quad x \in [0, b] \tag{1}$$

where μ is the traffic rate parameter, and b is the bound parameter which can be used to make the generated values bounded over the interval $[0, b]$ as the exponential distribution is theoretically unbounded.

Moreover, we adopted a randomized attack model, where malicious nodes decide to forward or drop each received packet randomly. In TrustMod, all values are evaluated automatically during the simulation process, while assessing the trust values without the TrustMod module is done offline. Different factors impact the dynamic evaluation of TrustMod, such as scheduled events processing and observation recording. Therefore, in order to validate the TrustMod module, the simulation scenarios will be run multiple times, and then the results will be averaged out. According to the Central Limit Theorem (CLT), a sample size of 30 is sufficient to estimate the Gaussian distribution [21]. Hence, we run the simulation 30 times for each simulation parameter settings followed by a statistic test. The two-sample Kolmogorov-Smirnov (K-S) test, which is a nonparametric hypothesis test, is applied to ensure that the two distributions obtained from the results using TrustMod module and trace file analysis are the same [22]. The K-S test rejects the null hypothesis at the 5% significance level. Table I shows the simulation parameters of LTMS. The malicious activities are integrated into the RouteInput method of the AODV protocol. The indirect trust evaluation has been manually stopped before the simulations in order to neutralize the recommendation exchange messages on the results. The trust scheme robustness against on-off attacks is evaluated using the on-off Attack Detection Metric proposed in [18]. The validation process is carried out for the following scenarios.

TABLE I: LTMS Simulation Parameters

Parameter	Value
Application	Poisson random traffic
Exponential transmission interval μ	5, 10, 20, 40
Packet size	264B
Routing Protocol	AODV (modified version)
Radio Range	1m
Propagation delay model	Constant speed propagation delay
Propagation loss model	Range propagation loss
Time unit	1s
Simulation Time	200s

A. Variable Traffic Rates

In this experiment, we validate our proposed trust module for variable traffic rates ranging from low to high. The traffic rate has been doubled each time, starting at $\mu = 5$. The obtained results using TrustMod and the offline analysis are averaged out and reported with 1 standard deviation in Fig. 7a-7d, which show the detection performance for variable on-off attack cycles. The figures show an identical behavior without any noticeable difference between using TrustMod and without using it. This ensures the design and implementation accuracy of TrustMod. The detection performance ranges between around 74% and 97%. The table II shows the two-sample K-S test results for each parameter settings, including the test decision for the 30 runs of simulation, P-value and test statistic. The minimum, maximum, mean and standard deviation values have been reported. It is worth mentioning that due to the high randomness level of the simulation settings, the results show some low minimum P-values. Therefore, the mean and standard deviation of the P-value have been reported to show that even though some low minimum values are shown in the table, the averaged P-value is still high, proving that the obtained trust values by the two methods are almost identical. The test statistic represents the maximum absolute difference between the two cumulative distribution functions as shown in Eq. 2.

$$D^* = \max_x(|\hat{F}_1(x) - \hat{F}_2(x)|) \tag{2}$$

where D^* represents the test statistic, $\hat{F}_1(x)$ is the cumulative distribution function for the trust values obtained using TrustMod modules, and $\hat{F}_2(x)$ is the cumulative distribution function for the trust values obtained using trace file analysis.

The results show that the average test statistic ranges between 0.029 and 0.064, which indicates that both cumulative distribution functions are very similar.

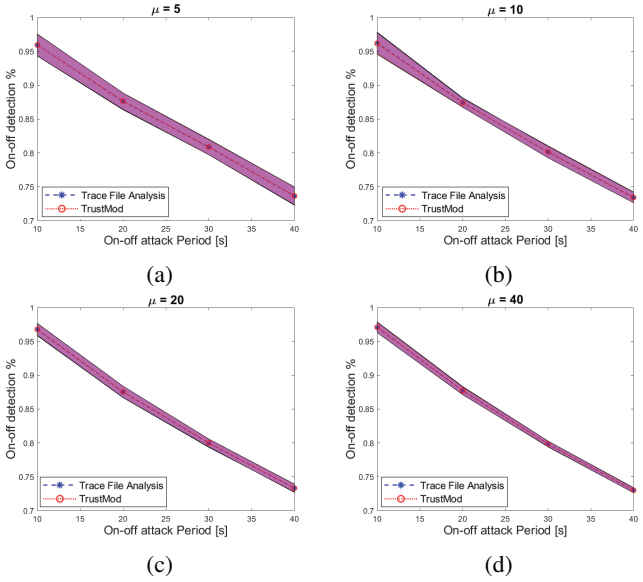


Fig. 7: The detection performance for variable traffic rates

B. Variable Drop Rates

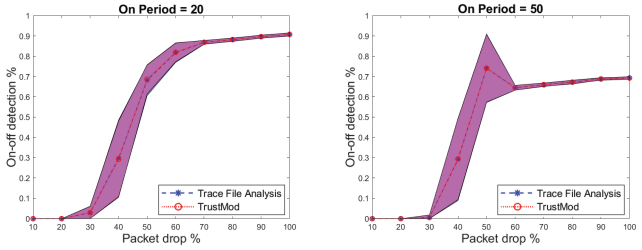
In this experiment, we use on-off attacks with variable drop rates to evaluate the detection performance using TrustMod and without using it. The drop rate varies between 10% and 100% during the on period. This experiment has been conducted for two different on periods, 20 and 50 time units using the same traffic rate $\mu = 10$. Fig. 8a and Fig. 8b show the detection performance with 1 standard deviation for both methods. It is obvious that the reported detection performance for TrustMod and trace file analysis are identical, which means that the two implementations follow the same behavior in attack detection.

On the other hand, a further investigation is done using the two-sample K-S test to ensure that the two obtained trust series come from the same distribution. Thirty simulation runs have

TABLE II: K-S test for variable traffic rates

Simulation parameters	K-S test decision	P-value				Test statistic			
		Min	Max	Mean	Std	Min	Max	Mean	Std
$\mu = 5$, $period = 10$	30/30	0.524	1.0	0.952	0.124	0.005	0.080	0.032	0.020
$\mu = 5$, $period = 20$	30/30	0.601	1.0	0.973	0.078	0.005	0.076	0.031	0.018
$\mu = 5$, $period = 30$	30/30	0.687	1.0	0.966	0.082	0.005	0.071	0.032	0.018
$\mu = 5$, $period = 40$	30/30	0.847	1.0	0.986	0.039	0.015	0.061	0.032	0.013
$\mu = 10$, $period = 10$	30/30	0.524	1.0	0.918	0.135	0.015	0.08	0.042	0.020
$\mu = 10$, $period = 20$	30/30	0.776	1.0	0.985	0.049	0.01	0.065	0.029	0.014
$\mu = 10$, $period = 30$	30/30	0.607	1.0	0.980	0.081	0.010	0.075	0.029	0.013
$\mu = 10$, $period = 40$	30/30	0.693	1.0	0.977	0.062	0.015	0.070	0.034	0.014
$\mu = 20$, $period = 10$	30/30	0.374	1.0	0.963	0.117	0.010	0.090	0.036	0.016
$\mu = 20$, $period = 20$	30/30	0.445	1.0	0.946	0.137	0.010	0.085	0.038	0.017
$\mu = 20$, $period = 30$	30/30	0.693	1.0	0.967	0.073	0.010	0.070	0.038	0.014
$\mu = 20$, $period = 40$	30/30	0.310	1.0	0.924	0.154	0.015	0.095	0.043	0.019
$\mu = 40$, $period = 10$	30/30	0.205	1.0	0.748	0.254	0.030	0.106	0.064	0.020
$\mu = 40$, $period = 20$	30/30	0.312	1.0	0.798	0.193	0.025	0.095	0.059	0.018
$\mu = 40$, $period = 30$	30/30	0.205	1.0	0.793	0.229	0.030	0.106	0.061	0.018
$\mu = 40$, $period = 40$	30/30	0.203	1.0	0.776	0.246	0.030	0.106	0.061	0.020

been carried out for each simulation parameter settings. This results in a total of 600 simulation runs. Table III shows the two-sample K-S test for the variable drop rates experiment. The null hypothesis has been accepted in 596 of 600 simulation runs, which makes up 99.3% of all simulation runs. The four rejected cases have been intensely investigated to find out the reason for the rejection. The comparison of the two trust series shows very close trust values over time, following the same trend. Moreover, the detection performance for both simulations was identical, which proves that both trust series are almost identical. Furthermore, the averaged P-value of the parameter settings that reported one rejection shows a high probability to accept the null hypothesis up to 92.5%. Therefore, this trivial difference in the trust values obtained from TrustMod module and trace file analysis can be attributed to the high simulation randomness and the system design. All generated traffics and packets dropping are randomly achieved during the simulation, as discussed earlier. Moreover, we obtain an efficient method to store and process the observations during the simulations. All TN objects are created just after the watchdog unit reporting new observations during a specific time unit. Afterward, all required processing is scheduled to be run at the end of the current time unit. This efficient mechanism may lead to having few observations reported in the next time unit as all processing is running on-line, in contrast to trace file analysis where the processing is running off-line after the simulation.



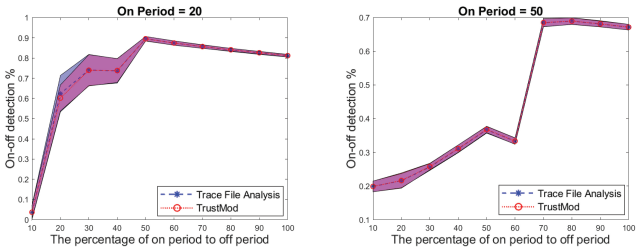
(a) (b)

Fig. 8: The detection performance for variable drop rates

C. Non-Identical Periods

In this experiment, we run more sophisticated on-off attacks to validate the obtained results of our TrustMod module. The ratio of the on-to-off period varies, starting from 10% to 100% in order to make the attack harder to detect. The traffic rate μ is set to 10, and two on periods have been considered 20 and 50. A total of 600 simulation runs for 20 different parameter settings are carried out. Fig. 9a and Fig. 9b show the detection performance with 1 standard deviation for both methods, TrustMod module and trace file analysis. The two figures show an almost identical performance for the two methods in both periods.

The same nonparametric hypothesis test has been used to ensure that both trust series come from the same distribution. Table IV shows the two-sample K-S test results. The test decision has accepted the null hypothesis in the 600 simulation runs, which makes up a 100% accepting rate. The averaged P-value ranges between 0.862 and 0.992, with a maximum P-value of 1 for all parameter settings. Moreover, the averaged test statistic shows very low values range from 0.027 to 0.050. These reported values, along with the detection performance results, prove that both trust series are almost identical, which ensuring the design and implementation accuracy of TrustMod.



(a) (b)

Fig. 9: The detection performance for different on-off ratios

VI. Performance Evaluation

In this section, we evaluate the performance overhead introduced by using the TrustMod module. A number of

TABLE III: K-S test for variable drop rates

Simulation parameters	K-S test decision	P-value Min	P-value Max	P-value Mean	P-value Std	Test statistic Min	Test statistic Max	Test statistic Mean	Test statistic Std
$\mu = 10$, $period = 20$, $drop = 10\%$	30/30	0.102	1.0	0.590	0.305	0.035	0.121	0.077	0.023
$\mu = 10$, $period = 20$, $drop = 20\%$	30/30	0.445	1.0	0.906	0.138	0.035	0.085	0.051	0.013
$\mu = 10$, $period = 20$, $drop = 30\%$	29/30	0.002	1.0	0.925	0.223	0.010	0.186	0.044	0.032
$\mu = 10$, $period = 20$, $drop = 40\%$	29/30	0.007	1.0	0.868	0.289	0.015	0.166	0.046	0.034
$\mu = 10$, $period = 20$, $drop = 50\%$	30/30	0.061	1.0	0.938	0.186	0.015	0.131	0.037	0.023
$\mu = 10$, $period = 20$, $drop = 60\%$	30/30	0.165	1.0	0.940	0.200	0.015	0.11	0.036	0.022
$\mu = 10$, $period = 20$, $drop = 70\%$	30/30	0.445	1.0	0.949	0.116	0.010	0.085	0.033	0.021
$\mu = 10$, $period = 20$, $drop = 80\%$	30/30	0.693	1.0	0.981	0.061	0.005	0.070	0.0271	0.016
$\mu = 10$, $period = 20$, $drop = 90\%$	30/30	0.607	1.0	0.969	0.099	0.005	0.075	0.030	0.017
$\mu = 10$, $period = 20$, $drop = 100\%$	30/30	0.310	1.0	0.943	0.150	0.005	0.095	0.033	0.021
$\mu = 10$, $period = 50$, $drop = 10\%$	30/30	0.080	1.0	0.582	0.279	0.030	0.126	0.077	0.022
$\mu = 10$, $period = 50$, $drop = 20\%$	30/30	0.445	1.0	0.884	0.155	0.025	0.085	0.052	0.015
$\mu = 10$, $period = 50$, $drop = 30\%$	30/30	0.524	1.0	0.964	0.094	0.020	0.080	0.040	0.013
$\mu = 10$, $period = 50$, $drop = 40\%$	29/30	0.000	1.0	0.744	0.330	0.025	0.307	0.067	0.053
$\mu = 10$, $period = 50$, $drop = 50\%$	29/30	0.034	1.0	0.847	0.296	0.010	0.141	0.049	0.033
$\mu = 10$, $period = 50$, $drop = 60\%$	30/30	0.524	1.0	0.953	0.107	0.010	0.080	0.040	0.015
$\mu = 10$, $period = 50$, $drop = 70\%$	30/30	0.696	1.0	0.984	0.056	0.020	0.070	0.036	0.011
$\mu = 10$, $period = 50$, $drop = 80\%$	30/30	0.849	1.0	0.976	0.040	0.015	0.061	0.038	0.013
$\mu = 10$, $period = 50$, $drop = 90\%$	30/30	0.524	1.0	0.961	0.098	0.015	0.080	0.037	0.015
$\mu = 10$, $period = 50$, $drop = 100\%$	30/30	0.776	1.0	0.973	0.056	0.009	0.065	0.037	0.014

TABLE IV: K-S test for non-identical periods

Simulation parameters	K-S test decision	P-value Min	P-value Max	P-value Mean	P-value Std	Test statistic Min	Test statistic Max	Test statistic Mean	Test statistic Std
$\mu = 10$, $period = 20$, $ratio = 10\%$	30/30	0.165	1.0	0.862	0.221	0.020	0.111	0.050	0.022
$\mu = 10$, $period = 20$, $ratio = 20\%$	30/30	0.254	1.0	0.877	0.217	0.020	0.101	0.047	0.022
$\mu = 10$, $period = 20$, $ratio = 30\%$	30/30	0.374	1.0	0.936	0.161	0.015	0.090	0.040	0.019
$\mu = 10$, $period = 20$, $ratio = 40\%$	30/30	0.524	1.0	0.934	0.134	0.015	0.080	0.039	0.019
$\mu = 10$, $period = 20$, $ratio = 50\%$	30/30	0.061	1.0	0.920	0.205	0.015	0.131	0.038	0.025
$\mu = 10$, $period = 20$, $ratio = 60\%$	30/30	0.524	1.0	0.957	0.109	0.015	0.080	0.035	0.017
$\mu = 10$, $period = 20$, $ratio = 70\%$	30/30	0.852	1.0	0.992	0.028	0.010	0.060	0.027	0.012
$\mu = 10$, $period = 20$, $ratio = 80\%$	30/30	0.852	1.0	0.992	0.028	0.010	0.060	0.028	0.013
$\mu = 10$, $period = 20$, $ratio = 90\%$	30/30	0.607	1.0	0.965	0.096	0.010	0.075	0.034	0.017
$\mu = 10$, $period = 20$, $ratio = 100\%$	30/30	0.776	1.0	0.986	0.043	0.015	0.065	0.031	0.014
$\mu = 10$, $period = 50$, $ratio = 10\%$	30/30	0.445	1.0	0.890	0.149	0.020	0.085	0.050	0.016
$\mu = 10$, $period = 50$, $ratio = 20\%$	30/30	0.205	1.0	0.909	0.204	0.024	0.106	0.046	0.020
$\mu = 10$, $period = 50$, $ratio = 30\%$	30/30	0.445	1.0	0.921	0.153	0.015	0.085	0.045	0.017
$\mu = 10$, $period = 50$, $ratio = 40\%$	30/30	0.693	1.0	0.935	0.096	0.030	0.070	0.046	0.013
$\mu = 10$, $period = 50$, $ratio = 50\%$	30/30	0.310	1.0	0.886	0.191	0.020	0.095	0.049	0.019
$\mu = 10$, $period = 50$, $ratio = 60\%$	30/30	0.131	1.0	0.868	0.246	0.020	0.116	0.050	0.023
$\mu = 10$, $period = 50$, $ratio = 70\%$	30/30	0.374	1.0	0.932	0.147	0.020	0.090	0.043	0.017
$\mu = 10$, $period = 50$, $ratio = 80\%$	30/30	0.607	1.0	0.961	0.092	0.012	0.075	0.037	0.016
$\mu = 10$, $period = 50$, $ratio = 90\%$	30/30	0.607	1.0	0.952	0.107	0.010	0.075	0.037	0.017
$\mu = 10$, $period = 50$, $ratio = 100\%$	30/30	0.607	1.0	0.976	0.076	0.015	0.075	0.034	0.014

simulations are carried out to measure the computational time and the used memory by increasing the network size.

A. Simulation Scenario

The network topology plays a significant role when evaluating the performance. For instance, choosing randomized parameters, such as nodes' positions and traffic rates, affects the comparison process of simulations. Therefore, in order to make the comparison fair, the grid position allocation has been adopted and the Constant Bit Rate (CBR) is used to generate traffic. Table V shows the used simulation parameters. The network topology consists of rows of nodes, where one of them acts as a sink, and other nodes act as clients. The propagation delay is constant, and the propagation loss model depends on the distance between the transmitter and receiver with a view to minimizing their impacts on the performance measurements.

TABLE V: Simulation Parameters

Parameter	Value
Application	CBR
Interval	1s
Packet size	264B
Routing Protocol	AODV
Radio Range	10m
Propagation delay model	Constant speed propagation delay
Propagation loss model	Range propagation loss
Time unit	10s
Simulation Time	500s

B. Performance Results

The performance overhead introduced by TrustMod is evaluated for different network sizes. The number of nodes in the network is increased from 5 to 200 nodes. The processing time to run the simulation scenario, in addition to the memory usage, including physical and virtual memory, are evaluated before and after enabling the TrustMod module. The experiments are carried out on an Intel Core i5-8500T processor at 2.1GHz and 4GB RAM. For each parameter settings, we run the simulations 10 times. This took around 5 days of simulation running on the aforementioned computer configuration. The results are then averaged out and reported in the following figures considering 1 standard deviation. Fig. 10 shows the processing time of using the TrustMod module for different network sizes. It

shows a minimal increase in processing time for large network sizes along with unnoticeable overhead for small and medium networks, ensuring the lightweight design of TrustMod.

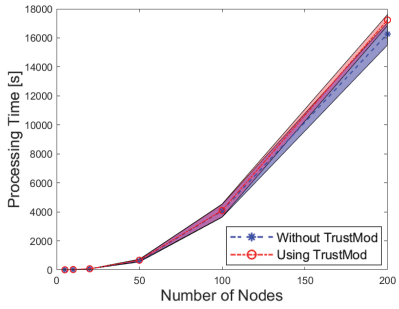
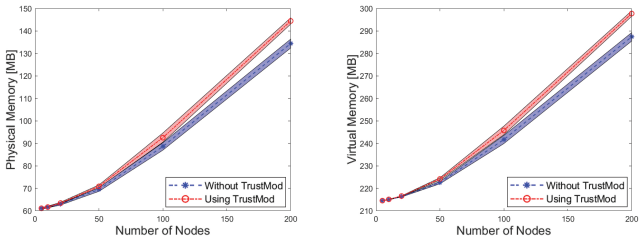


Fig. 10: The processing time

The second important metric is memory consumption. The average virtual and physical memory consumption are evaluated for a different number of nodes. Fig. 11a and Fig. 11b show the average consumed physical and virtual memory, respectively. The memory consumption is monitored during the simulation by scheduling five events with a lag that represents a 20% of the total simulation time. TrustMod module provides a memory conservative trust management testbed. The memory overhead is unobtrusive for small and medium network sizes. Moreover, it shows an increase of around 7% and 4% for physical and virtual memory for large networks, respectively. This slight increase is expected by increasing the network size because each node maintains the whole observations and trust evaluations for all nodes that were communicated during the simulation.



(a) (b)

Fig. 11: Memory Consumption

VII. Conclusion and Future Work

Trust management schemes provide a powerful tool to reinforce the overall security. Our proposed trust management module, TrustMod, provides a lightweight and accurate testbed to evaluate the effectiveness of the proposed trust schemes. TrustMod is designed to be an NS-3 module and can be easily integrated and used with any network protocol. Validation results prove the high accuracy of the trust evaluations for a wide range of simulation parameter settings with minimum resource overhead. In the future, the uncertainty and indirect trust components will be validated using various proposed methods in the literature.

References

[1] K. Govindan and P. Mohapatra, "Trust computations and trust dynamics in mobile adhoc networks: A survey," *IEEE Communications Surveys Tutorials*, vol. 14, no. 2, pp. 279–298, 2012.

[2] H. Xia, S. Zhang, Y. Li, Z. Pan, X. Peng, and X. Cheng, "An attack-resistant trust inference model for securing routing in vehicular ad hoc networks," *IEEE Transactions on Vehicular Technology*, vol. 68, no. 7, pp. 7108–7120, 2019.

[3] M. S. Hajar, M. O. Al-Kadri, and H. K. Kalutarage, "A survey on wireless body area networks: Architecture, security challenges and research opportunities," *Computers & Security*, p. 102211, 2021.

[4] Open source, "Ns-3 a discrete-event network simulator for internet systems," accessed: 01-04-2020. [Online]. Available: https://www.nsnam.org/releases/

[5] D. Pal, "A comparative analysis of modern day network simulators," in *Advances in Computer Science, Engineering & Applications*. Springer, 2012, pp. 489–498.

[6] Y. Zhang, W. Wang, and S. Lü, "Simulating trust overlay in p2p networks," in *Int. Conf. on Computational Science*. Springer, 2007, pp. 632–639.

[7] F. G. Mármol and G. M. Pérez, "Trmsim-wsn, trust and reputation models simulator for wireless sensor networks," in *2009 IEEE International Conference on Communications*. IEEE, 2009, pp. 1–5.

[8] NS-2, "Network simulator ns-2," accessed: 25-04-2019. [Online]. Available: https://www.isi.edu/nsnam/ns/

[9] M. Lacage and T. R. Henderson, "Yet another network simulator," in *Proceeding from the 2006 workshop on ns-2: the IP network simulator*, 2006.

[10] G. F. Riley, "Large-scale network simulations with gtnets," in *Winter Simulation Conference*, vol. 1, 2003, pp. 676–684.

[11] J. Lakkakorpi and P. Ginzboorg, "ns-3 module for routing and congestion control studies in mobile opportunistic dtns," in *2013 International Symposium on Performance Evaluation of Computer and Telecommunication Systems (SPECTS)*. IEEE, 2013, pp. 46–50.

[12] G. Carneiro, P. Fortuna, and M. Ricardo, "Flowmonitor: a network monitoring framework for the network simulator 3 (ns-3)," in *Proceedings of the Fourth International ICST Conference on Performance Evaluation Methodologies and Tools*, 2009, pp. 1–10.

[13] V. B. Reddy, S. Venkataraman, and A. Negi, "Communication and data trust for wireless sensor networks using d–s theory," *IEEE Sensors Journal*, vol. 17, no. 12, pp. 3921–3929, 2017.

[14] N. Labraoui, M. Gueroui, and L. Sekhri, "A risk-aware reputation-based trust management in wireless sensor networks," *Wireless Personal Communications*, vol. 87, no. 3, pp. 1037–1055, 2016.

[15] D. He, C. Chen, S. Chan, J. Bu, and A. V. Vasilakos, "Retrust: Attack-resistant and lightweight trust management for medical sensor networks," *IEEE transactions on information technology in biomedicine*, vol. 16, no. 4, pp. 623–632, 2012.

[16] W. Zhang, S. Zhu, J. Tang, and N. Xiong, "A novel trust management scheme based on dempster–shafer evidence theory for malicious nodes detection in wireless sensor networks," *The Journal of Supercomputing*, vol. 74, no. 4, pp. 1779–1801, 2018.

[17] J. Zhao, J. Huang, and N. Xiong, "An effective exponential-based trust and reputation evaluation system in wireless sensor networks," *IEEE Access*, vol. 7, pp. 33 859–33 869, 2019.

[18] M. S. Hajar, M. O. Al-Kadri, and H. Kalutarage, "Ltms: A lightweight trust management system for wireless medical sensor networks," in *The 19th IEEE International Conference on Trust, Security and Privacy in Computing and Communications (TrustCom 2020)*, 2020.

[19] W. Fang, C. Zhu, W. Chen, W. Zhang, and J. J. Rodrigues, "Bdtms: Binomial distribution-based trust management scheme for healthcare-oriented wireless sensor network," in *2018 14th International Wireless Communications & Mobile Computing Conference (IWCMC)*. IEEE, 2018, pp. 382–387.

[20] M. S. Hajar, M. O. Al-Kadri, and H. Kalutarage, "Etaree: An effective trend-aware reputation evaluation engine for wireless medical sensor networks," in *2020 IEEE Conference on Communications and Network Security (CNS)*. IEEE, 2020, pp. 1–9.

[21] H.-J. Chang, C.-H. Wu, J.-F. Ho, and P.-y. Chen, "On sample size in using central limit theorem for gamma distribution," *Information and Management Sciences*, vol. 19, no. 1, pp. 153–174, 2008.

[22] Y. Xiao, "A fast algorithm for two-dimensional kolmogorov–smirnov two sample tests," *Computational Statistics & Data Analysis*, vol. 105, pp. 53–58, 2017.