Journal of Ubiquitous Systems & Pervasive Networks
Volume 3, No. 1 (2011) pp. 00-00

A Novel Micro-services Cluster Based Framework for Autonomous Vehicles

Henry Alexander Ignatious[a], Hesham-El-Sayed[a], Manzoor Khan[a], Shawqi Kharbash[a] and Sumbal Malik[a]

[a]*College of Information Technology, United Arab Emirates University, Al Ain, UAE*

Abstract

Modern technologies like digital spaces, intelligent transportation, and digital operations are faced with technical challenges related to data capturing, data processing, data storage, data security, communication, etc. Ensuring security and reliable message propagation among autonomous vehicles is a major challenging task. Establishing appropriate environment for developing vehicular-based solutions by the vehicle manufactures increases their estimated cost and time. Hence, to minimize this problem, this proposal aims to introduce Service Oriented Architecture (SOA) principles, to access trust based solutions as simple cloud based services by the clients. We propose a framework to integrate three mandatory vehicular applications namely trust estimation, secured message dissemination, and routing as cloud-based microservices. We also propose an innovative CipherText Policy Attribute Based Encryption (CP-ABE) algorithm to ensure confidentiality of data by an access control system in highly dynamic and automated network. The services are deployed as Docker images using advanced concepts of Dockers and Containers. Dockers coordinate the orchestration of multiple tasks related to the proposed microservices and help to implement the services in cross-platform environments. These services can be implemented in both autonomous and manual vehicular systems. The service providers can charge the clients based on the usage of the services. A detailed experimental analysis is accomplished to evaluate the performance of the proposed micro services in cross platform environments; further, an extensive simulation is performed to assess the individual performance of the proposed vehicular applications.

Keywords: *Vehicular Networks, Trust Management, Cloud Computing, Dockers &Containers.*

1. Introduction

Road safety, congestion control and security metrics are the three important factors to be considered, while regulating the traffic system. Traditional traffic controlled systems are slowly replaced by state-of-the-art vehicular technologies like vehicular networks, intelligent roadside units, Artificial Intelligence based decision mechanisms for critical maneuvers, etc. The concept of connected vehicles is known and is being implemented by automobile makers allowing the vehicles to exchange messages. Hence, ensuring trustworthiness among the vehicles within the specified range, during message interchange to maintain integrity, is an important task. Moreover, data transmission between the vehicles directly affects the security in vehicular environment. The quality of safety or non-safety applications in vehicular network, largely depend on the trustworthiness of the data. Trust plays a vital role in security and quality of vehicular network [1]. Automated vehicle systems, fully autonomous vehicles, and alternative transportation services (ride sharing, car sharing, etc.) are now constantly in the news. A range of Tier-1 providers, automobile makers, equipment manufacturers, startups and academic organizations are leading various technological efforts to develop the systems necessary to make transportation more responsive, accessible and ultimately safer for all consumers across various communication generations [2]. In future, the autonomous vehicles will gradually replace the manual driven vehicles and fundamentally change all road traffic conditions. The main objective of autonomous driving is ensuring safety and security of the people in public roads. Human drivers due to many external factors cause most of the road accidents. Whether introducing full-fledged autonomous driving reduces accidents and ensures safety and security among the people in public roads is still a debating topic between the manufactures and the researchers. Moreover, the usage of autonomous driving is restricted in many countries due to various legal issues [3].

There are many challenges associated with launching a complete autonomous vehicle. The major challenges include the understanding of the instant and dynamic behavior of road conditions, traffic conditions, accident liability, radar inferences, establishing trust between the vehicles, dissemination of reliable information among the vehicles and weather conditions. The foremost problem for the manufacturers and researchers is how to make the autonomous vehicle behaves like humans in dynamic decision-making. The autonomous vehicles are collection of intelligent sensors, actuators, sophisticated algorithms and powerful processors to excite the software and physically controls the vehicles. Recent statistics suggests that twenty-five countries are already prepared to implement autonomous driving. According to official reports, the UAE is expected to finalize autonomous vehicle standards and regulations in 2020, and accordingly autonomous vehicles may enter the UAE market within one year afterwards by 2021 [4].

However, there are many existing challenges associated in launching autonomous vehicles, the key and the mandatory challenge to be addressed is ensuring safety and security among the AVs during their travel. Hence, establishing trust between the AVs and ensuring reliable message dissemination between the AVs are

* Corresponding author. Tel.: +971543781405
E-mail: 201990006@uaeu.ac.ae

DOI: 10.5383/JUSPN.03.01.000

mandatory tasks. Hence, more innovative studies and contributions are needed to develop versatile solutions to monitor and control the overall activities of autonomous driving. The data acquired from the sensors plays a vital role for controlling various activities of autonomous driving. Today's autonomous vehicles manufactures and the community, which lends autonomous driving for rental basis, do not spend their time and money to scale their solutions further. Instead, they extend the new services from third party web services, which are deployed in cloud environments. Hence, in today's scenarios, many micro services are developed for various applications representing multiple domains. The concepts of micro services reduce various complexities related to large applications and helps several organizations to minimize their cost and time to develop and maintain complex applications [5].

However, there are many existing challenges associated in launching autonomous vehicles, the key and the mandatory challenge to be addressed is ensuring safety and security among the AVs during their travel. Hence, establishing trust between the AVs and ensuring reliable message dissemination between the AVs are mandatory tasks. Hence, more innovative studies and contributions are needed to develop versatile solutions to monitor and control the overall activities of autonomous driving. The data acquired from the sensors plays a vital role for controlling various activities of autonomous driving. Today's autonomous vehicles manufactures and the community, which lends autonomous driving for rental basis, do not spend their time and money to scale their solutions further. Instead, they extend the new services from third party web services, which are deployed in cloud environments. Hence, in today's scenarios, many micro services are developed for various applications representing multiple domains. The concepts of micro services reduce various complexities related to large applications and helps several organizations to minimize their cost and time to develop and maintain complex applications [5].

This paper proposes a new framework, which integrates three main modules namely a trust estimation module, a security module and a routing module.. Trust evaluation services estimates the trust between the vehicles and reliable messages in encrypted form are propagated between the trusted vehicles. Further the trust evaluation services integrates an innovative cluster algorithm to group the vehicles and an innovative CipherText Policy Attribute Based Encryption (CP-ABE) algorithm to ensure confidentiality of data by an aces control system in highly dynamic and automated network. The routing service reroutes the AVs if they face any unexpected roadside events like accidents (or) congestions. The users are charged based on the usage of the services. The study is planning to provide the above-mentioned applications as services to the autonomous vehicle manufacturers and autonomous vehicle hiring community, to minimize their time and cost effectively. Subsequently the service providers will be benefitted by charging the autonomous vehicle manufacturers and other hiring community for sharing their services. An innovative methodology has to be selected to convert the proposed services as platform independent applications. Hence, this study selected the Docker concepts to deploy the proposed services. The Docker concepts provides many advanced features, which facilitates the developers to deploy their applications as services in public registries. The Dockers also helps to convert the services as platform independent applications. There is no need to integrate a virtual operating system along with the service, to execute them in cross platforms. This strategy minimizes the development of individual services to serve different operating systems, thus reducing the time and cost of the service developers.

Applications are developed for the suggested services and apparently, the services are deployed as a Docker images. The clients can extend the Docker image as a cloud-based web service and integrate them in their existing applications. So far, very few studies and contributions have been proposed to implement of micro services in autonomous driving to effectively maintain and monitor them.

The paper is organized in the following hierarchy. Section, two briefs the background of this study, Section three, discusses the existing literature on various areas related to vehicular networks like trust management, security issues and routing followed by a brief review about designing and implementing the microservices.. Section four, discusses the overall functionality of the proposed framework, along with the steps involved in deploying and executing Docker images. Section five briefs illustrates various experimental and comparative analyses, Section six describes the future enhancements and Section seven concludes the paper with a short summary of this study.

2. Background

For the current and envisioned application scenarios, cloud based solutions are in more demand due to many advantages such as cost saving, scalability, platform independency, high speed, easy integration more virtualization, reliability, security and so on. Cloud platforms follows Service Oriented Architecture (SOA), where the applications are shared among the users as demand based services. It is an architectural style used to segment complex applications into individual modular services, which are easy to maintain, tested, loosely coupled, and independently deployable. The micro service architecture enables the rapid, frequent and reliable delivery of larger complex applications.

Each service performs a unique function. All services must logically represent a repeatable business activity with a specific outcome. They must be self-contained. Its operation must be abstract to the users (its implementation part must be hidden to the users). A service can be composed of other services. Micro services are extensions of SOA based services. They almost extent the characteristics of the web services. Most of the organizations and other application developers integrate cloud-based microservices due to the following advantages like high maintainable and testable, loosely coupled, independently deployable, organized around business capabilities and owned by a small team. The microservice architecture enables massive, complex applications to be delivered quickly, frequently, and reliably. It also allows a company's technology stack to grow. Moreover, the microservices are relatively small and the user can easily understand their behavior. The application starts up faster, allowing developers to be more efficient and release times to be reduced. Microservice architecture supports fault isolation, where the effected service is isolated, allowing other services to operate without interrupting the performance of the overall system. All the microservices can communicate between the users using the standard web based protocols like http. Figure 1, illustrates the functionality of microservices. Services are developed and deployed in public registries present in the web servers. The users can access and communicate with the services through standard APIs and web based protocols like http.

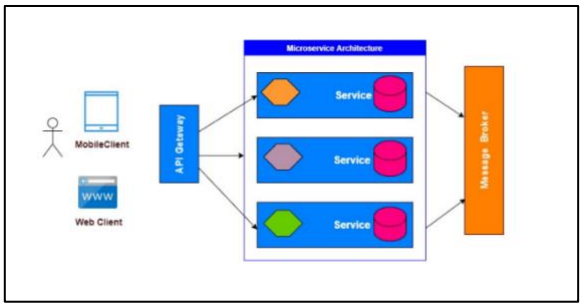

Fig. 1. Microservice Architecture

This study uses Google cloud based services to develop and deploy the proposed applications as microservices. The following subsections briefs the salient features of the Google cloud based components, which are used to deploy the microservices.

2.1. Google Cloud

Google Cloud Platform (GCP), is an important service offered by the Google platform just as other services like Gmail, Google Search, and YouTube, which serves the end-users. Apart from the management services, GCP provides modular based cloud services like computing, data storage, data analytics and machine learning. Currently Google lists over 90 cloud services under Google Cloud Brand. Key services such as Cloud AI, API platforms services, and IoT, are used to develop scalable Artificial Intelligence applications, IoT based services and versatile application program interfaces to communicate with multiple applications that executes in multiplatform environments.

2.2. Dockers & Containers

Docker is a platform for developers and system admins to develop, ship and execute the applications in different cross platforms. Docker is a set of platforms as a service (PaaS) products that uses OS-level virtualization to deliver software in packages called containers. The main components of Dockers are Docker Image, Docker Container, Docker Engine and Docker File. Containers are applications which are used to package the application images and help to execute them in cross-platform environments [15][16][17]. Figure 2, illustrates the Docker architecture. The major components of the Docker are (i) Docker Engine, (ii) Docker Client (iii) Docker Registries and Docker Objects.

The Docker Engine is the heart of the Docker architecture. It is installed in the host machine. It follows client/server architecture and has three major components namely a server, which is the Docker daemon used to create and manage Docker images, a REST API, which coordinates the activities of the Docker daemon and Command Line Interface (CLI), used by the client to enter Docker commands. The next component namely the Docker client, who interacts with the server to access the Docker images. Followed by the Client component is the Docker registry, a location in the server to store the Docker images. The Docker hub can be either a public (or) private. The last component is the Docker objects, which contains images, containers, volumes and networks. Docker images are read only templates used to create Docker containers. Docker containers are objects which stores the instances of the applications along with their environmental information used by the clients to execute in their local machines.. The persistent data created by accessing the Dockers by the clients are organized and stored in volumes, which are maintained by the servers. Finally, the Network object contains drivers, which helps the client to interact with the Docker objects. Network object consists of five important drivers namely bridge, host, overlay, none and macvlan. Bridge is a driver to container, which is used when multiple client containers communicate with the host. Host removes the network isolation between Docker containers and Docker host. Overlay driver enables swarm services to communicate with each other. None driver is used to disable all networking services. Finally, macvlan driver is used to assign mac addresses to the containers to make them look like physical devices.

Once when a client runs the Docker run command to access the web services, initially a Docker build command is triggered, which creates a Docker template, locates and pulls the specified application image from the Docker hub. The images are further packaged in the Docker container, which will be executed in the client's local machine.

3. Related Work

The basic concept of road vehicle automation refers to the replacement of some or all of the human labor of driving by electronic and/or mechanical devices. Origins of the automated driving technology can be traced back to the early 20th century. At that time, the technology was concentrated on autonomous speed, break, lane control, and other basic cruise control aspects. However, only during the last decade or so, incubating conditions of the Digital and 4th Industrial Revolutions gave birth to rapid technological advancements in the field; resulting in numerous prototype Autonomous AVs being trailed on the roads. Many research articles have been published in the academic literature describing the technological advancement of AVs.

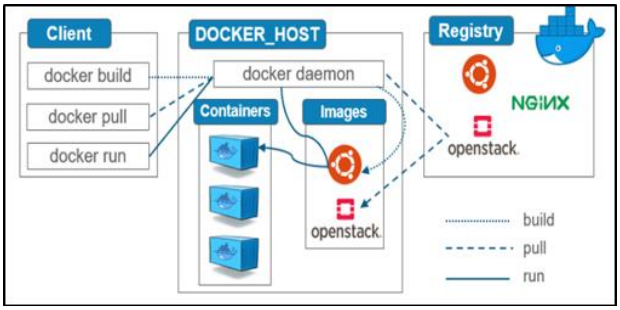

Fig.2. Docker Architecture

Recent studies done by El-Sayed et al [6], have proposed a novel direct trust evaluation method to estimate trust between the vehicles based on their behavior and message propagation rate using Bayes theorem concepts. Similarly innovative studies done by Gianmarco Baldini et al [7], have proposed a trust estimating technique using Block Chain concepts. Vehicles are issued with a valid Public Key Certificate (PKI) and further allowed to propagate messages to other vehicles, if they satisfy some criteria stored in the blocks. Siri Guleng et al [8], have proposed a direct trust estimating technique using fuzzy logic to calculate trust between the trustee nodes within transmission range of trustor nodes. Subsequently they have also implemented reinforcement-learning techniques to calculate indirect trust among the vehicles where direct trust is not possible.

Jannik Lotz et al [9], have conducted an exclusive survey to analyze the feasibility and after effects of converting complex software-based driver assistance system to micro service architecture. The complex software-based driving assistance system is segmented into individual micro services, addressing a particular problem of driving assistance system. The authors have

discussed a micro service architecture, which assists the autonomous vehicles in lane following system. Similarly, Duo Lu et al [10] have designed and developed a framework to distribute various functionalities of IoT based solutions as micro services. A micro service approach to build an IoT system can combine a mutually enforcing way with patterns for micro services, API gateways, distribution of services, uniform service discovery, containers, and access control. The authors' study gave a wide scope and knowledge to understand the basic concepts, implementation and deployment of micro services. The authors have also have proposed a service-oriented framework which integrates the platooning functionalities as micro services. The micro service utilizes other software modules of the proposed framework for their effective functioning. The authors tested the framework with low cost robots. Their hope is that the platooning as a service approach can help in the construction of more efficient, interoperable, and secure autonomous vehicles in the future. The author's contribution further guided and helped to design, develop and deploy the micro services.

Muhammad Alam et al [11] have proposed a modular and scalable architecture based on lightweight virtualization. The provided modularity, combined with the orchestration supplied by Docker, simplifies management, enables distributed deployments and creates a highly dynamic system. Similarly, Qian Qu et al [12], have investigated multiple micro service deployment policies on edge computing platform. The micro services are developed as Docker containers, and comprehensive experimental results demonstrate the performance and interference of micro services running on benchmark scenarios. Studies done by Aravinthan Gopalasingham et al [13] presented the virtualization of a prototyped software, which defined radio access network (RAN) architecture by using VMs and Docker containers. In addition, it provides an analytical model for the generalized software defined RAN architecture with the practice of VM based and Docker container-based implementations. An innovative research done by Jose Ramirez et al [14], explored these issues by looking at a reference architecture for services and developed a practical framework for service management in vehicular networks. A functional system looking at migration of a Network Memory Server using different migration techniques such as Docker, KVM, LXD and Unikernels is explored and the presented results showed that the authors' proposed approach can be used to support new applications and services in a highly mobile environment.

To secure user identities, the schemes in [26] suggested using Public Key Infrastructure (PKI). To reduce the communication overhead of message transmission on VANET, Prema suggested homomorphic encryption [27]. for VANET, Gao and Xin suggested a new location privacy scheme [28]. The authors recommend using a group of key encryptors to establish an encrypted area when the vehicle/vehicles need to change their pseudonyms. Duan et al using a Software-Defined Network (SDN) powered vehicle [29] proposed the adaptive vehicles cluster. SDN distributes the data, and the vehicle enters the cluster adaptively based on the data traffic. To solve the scalability, connectivity, and flexibility problems in the VANET method, Truong et al proposed a hybrid (of Fog and SDN) scheme [30]. [31] Suggested a VANET-based algorithm for constructing the optimal service function chain cluster (SFC).

In current scenarios, applications developed must be generic that can be implemented in any cross platform environment. Since most of the vehicular solutions are developed and implemented using different tools, executing in multiple platforms, our developed services must be compatible with all vehicular applications developed under various conditions. Further, the services must be easily accessible, secured, scalable and deployed effortlessly, to enhance the client's requirement. Dockers and containers satisfy all the above-mentioned requirements. Containers are used to package the developed services and Dockers, utilizes the client's operating system kernel, to convert the services compatible with any cross platform. Moreover, Dockers and Containers supports cloud-architect. Current service architectures do not fully provide such features and hence a new service management framework for vehicular networks has to be initiated. To the best of our knowledge, most of the current studies mentioned micro services are at a conceptual level. In this work, we take this forward by highlighting an implementation with emphasis on infotainment as a use case. This study uses an algorithmic approach, which is detailed, in the ensuing sections.

4. Proposed framework and services

In this paper, we propose a framework to integrate three modules, which are then transformed into microservices. The first module estimates the trust between the vehicles and the second module focuses on the security issues and the third module emphases on routing activities. Later these applications are converted into microservices with the help of Dockers. The application images are deployed in cloud based Docker registries. Containers are used to deploy, package and ship the micro service to be executed in cross-platform environments. Based on the user requirement single or multiple image instances representing different application are packaged using the Containers and subsequently executed in the client's machine. The Docker/Container services use the client's OS kernel to convert the application image to execute in cross platform environments. The autonomous vehicle manufacturers and solution developers can extend the Docker images as simple web services in their existing (or) developing applications, which minimizes their cost and time in developing complex applications.

Figure 3, illustrates the overall flow of the proposed architect. The first step is developing the three modules and integrating them in the proposed framework. The second step is to convert the modules as Docker images and deploy them in a public Docker hubs (or) registries located in cloud-based servers. The third step is accessing the services by the Trust Management Services present in the RSUs. The fourth step is recording of roadside events in the RSUs by a trusted vehicle. The fifth step is estimation of trust for other neighboring vehicles by the proposed Trust Management Service TMS. The sixth step is dissemination of encrypted message along with a secret key to the trusted vehicle, which in turn propagates the encrypted message along with the secret key to other trusted vehicles in the cluster [19].

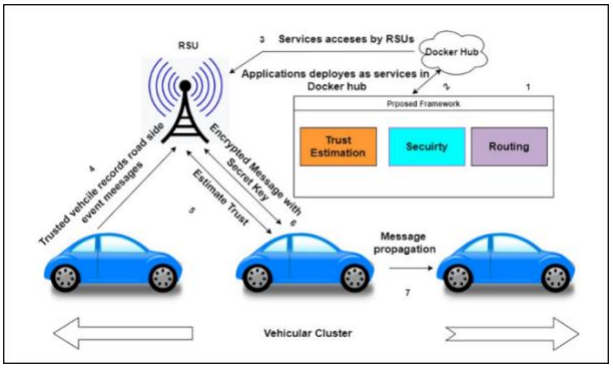

Fig 3. Overall flow of the proposed framework

4.1. Proposed clustering algorithm

The study proposes an entity based indirect trust evaluation scheme to estimate the trust of the vehicles. The TMS evaluates the trust of every individual vehicle within the RSU range using the suggested trust evaluation method. The reputation score is calculated using the following Eq 1.

$$\boldsymbol{Trust_{val}} = (\boldsymbol{Trust_{val}} \times \boldsymbol{(1-\alpha)} + \boldsymbol{LR} \times \propto) \quad \textbf{(1)}$$

Where (LR) is the last feedback rating and (α) represents a weight assigned to Eq.1., which normally represents values between the range [0, 1]. For trust estimation, the constant factor (α) is assigned with a value of (0.2). If any vehicle in the RSU range is a new entry and which does not have previous feedback value, the TMS collects valid information related to the new vehicle such as the register number and forwards the information to RTA. The RTA in turn validates the register number of the new vehicle and sends useful information like the owners name, the registration date, accident rate etc. related to the vehicle, to the TMS for further processing. For the new vehicles, the proposed TMS assigns an initial trust value of (0.5) value and subsequently the trust values are updated based on the vehicles behaviors. TMS assigns a value of (1) to the factor ($\boldsymbol{LR}$) in equation (1), for good behavior of vehicles, and a negative value say (-1) for bad behavior of vehicles. .Based on the trust scores and distance metrics a cluster is formed by the TMS using the suggested clustering algorithm depicted in Table 1. The TMS selects a vehicle with the maximum trust score as the cluster head. Encrypted messages along with the secret key (SK), key are first disseminated by the TMS to the cluster head, which in turn propagates to other vehicles within the cluster. Figure 4, illustrates the overall flow of the proposed framework [20].

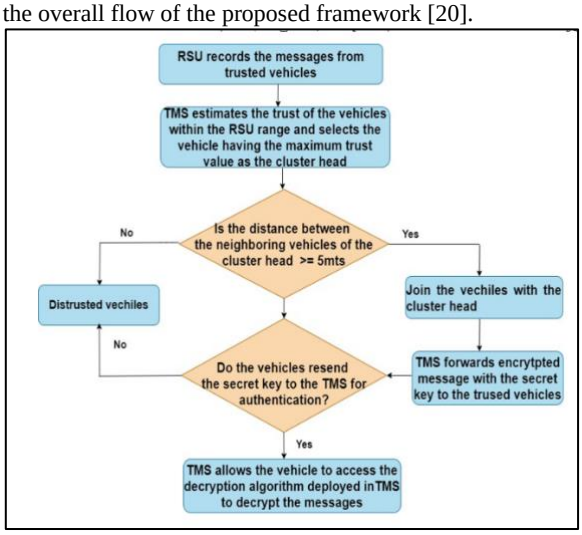

Fig. 4. Flow of the proposed framework

4.2. Proposed clustering algorithm

Table 1 below summarizes the proposed clustering algorithm, where the sequential steps are detailed.

Table 1: Proposed Clustering Algorithm

Step 1: TMS evaluates the trust score of all the vehicles within the RSU range.
Step 2: TMS selects the vehicle with the maximum trust score as the cluster head.
Step 3: TMS measures the Euclidean distance between the cluster head and its neighboring vehicles.
Step 4: If the distance is within an optimal value (5 mts), then the vehicle joins the cluster head.
Step 5: The procedure is repeated until all the vehicles within the RSU range are scanned.
Step 6: Remove any vehicle from the cluster, which violates the above-mentioned condition.

4.3. Message dissemination service

The focus of this study is ensuring secured and reliable message exchange between the vehicular entities and trusted vehicles. Hence, to solve this issue, this study has proposed an innovative encryption algorithm, which inherits some of the ideas from the famous CP-ABE encryption/decryption algorithm. The algorithm generates a secret key along with the encrypted Cipher text and propagates them to the trusted vehicle in the cluster. The trusted vehicle using the secret key decrypts the message using the sale algorithm. Secret key is generated using the following equation.

$$SK = rand(prime\ number) \times Length\ (M)^{\propto} \quad (2)$$

Where (M) denotes the message and $(\propto)$, signifies the weight of the message. All messages related to roadside events are assigned with a weight value of (2) and for other messages like advertisements, infotainment services etc., the weights are assigned with a value of (1). The encryption and decryption activities are performed by the suggested encryption/decryption algorithm following the below mentioned steps.

Step 1: Encryption (SK, M, A) - CT
Step 2: Decryption (CT, SK) – M

Where (SK) represents the secret key, (M) the original message, (A) denotes access control and (CT) is the CipherText
Table 2, illustrates the overall functioning of the suggested encryption/decryption algorithm. The encryption algorithm takes three input parameters namely secret key, message and access control. For trusted vehicles, the access control value is set to one, which highlights the vehicles have full privilege to access the secret key and decrypt the message. Other non-trusted vehicles are assigned with a value of zero, the vehicles can access the message, but they are not allowed to decrypt the message. Initially the decryption algorithm checks the access control value of the vehicles. If the value is one, then it converts each character of the message string with its ASCII value, and increments the value with one [32][33].

Table 2: Proposed Encryption/Decryption Algorithm

Encryption (SK,M,A)
Step 1: if (A=1)
Step 2: Read (M)
Step 3: While $(M[i] <> NULL)$
Step 4: $\mathrm{CT[i]} = \frac{\mathrm{ToAscii(M[i]+1)}^{(\mathrm{Length(M)})}}{\mathrm{Length(M)Div\ 2}}$
Step 5: $i = i + 1$
Step 6: Return (CT)

Decryption(CT, SK)
Step 1: Read(CT)
Step 2: $While\ (CT[i] <> NULL)$
Step 3: $M[i] = \frac{ToAscii(CT[i]-1)^{(Length(CT))}}{Length(CT)Div\ 2}$
Step 4: $i = 1 + 1$
Step 5: Return(M)

The value obtained is multiplied with the power of the message length. The results obtained are further divided by the (Div 2) operation performed over the message length. The encrypted message along with the secret key is forwarded to the trusted vehicles and the TMS prompts the vehicles to resend the secret key to the TMS. This step is performed to authenticate the trusted vehicles. The TMS crosschecks the secret ley received from the trusted vehicle and once the vehicle is authenticated, the TMS allows the vehicle to access the decryption algorithm from the TMS to know the actual message.

In decryption process, the only modification done is decrementing the ASCII value of individual characters in the Cipher text (CT) by a value of one. Remaining processes are same as the encryption algorithm to get the original message. Figure 5, illustrates the various stages involved in the proposed encryption and decryption algorithm.

4.4. Proposed routing service

Further, the study has also suggested a heuristic based routing service, which uses the concepts of Tabu search to effectively route the vehicles to their desired destination [21].

4.4.1. Functioning of proposed Routing service

Initially the Tabu search initiates with the starting city, intermediate cities and ending city along with the best routing solution. The maximization functioning is estimated using Eq.3. Average cost associated with the adjacent nodes of a randomly selected node are calculated. The adjacent node whose maximization function value is less than or equal to its original cost is selected as the next node for further processing by the current node. Subsequently the values in the Tabu list are updated and the value of the old best solution is updated with the new best solution. If the value of the maximization function calculated for the adjacent nodes of a random node is greater than (or) not approximately equal to the previous best solution, the nodes information is removed from the Tabu list and the procedure is repeated for the other nodes until the termination condition is reached. The nodes selected by the Tabu search along with their associated cost is the optimal path for that particular trip [21-22].

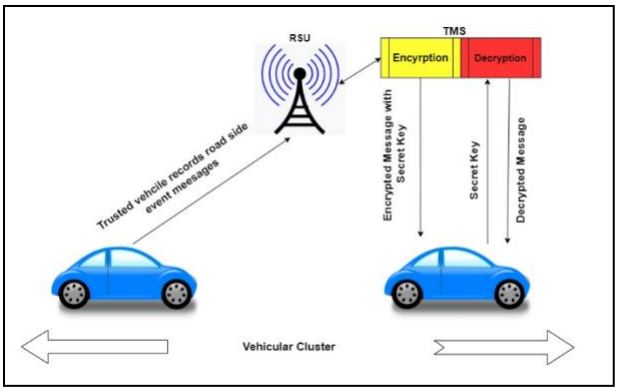

Fig. 5. Encryption/Decryption Process

Maximizing Function

$$MaxFunc_{TS} = \sum_{i=1}^{n} \frac{Cost_{(i)}}{n} \qquad (3)$$

Where $\boldsymbol{Cost}_{(i)}$, is the cost is associated with the adjacent node of a randomly selected node and (n) represents the total number of adjacent nodes of a selected node in a graph. If Eq.3, has it value less that (or) approximately equal to the cost of the adjacent nodes, that particular node is selected as the next node for further processing. The above statements are represented in Equation (4). Figure 6 illustrates the functional flow of the suggested routing algorithm [22 -23].

$$MaxFunc \leq Cost(i) \qquad (4)$$

4.5. Steps to deploy the services

The following section explains the steps involved in deploying and executing Docker images

Step 1: Create the new project in Google Cloud platform.
Step 2: Create the environment files required for the Docker image
Step 3: Develop the services.
Step 4: Build and deploy the services in local Google Container Register (gcr)
Step 5: Create a REST API
Step 6: Pull and push the images via REST API
Step 7: Execute the service in any platform.

4.5.1 Build and deploy images

Command - gcloud builds submit --tag gcr.io/stately-math-273218/my-python-

Run the Docker image

The Docker is executed in clients' container using the Docker run command [18]. Where --rm -p 8000:8000 is the port through which the application is communicating, gcr.io/stately-math-273218/my-python-app_test refers the google cloud register, project id and the image instance respectively.

Command - docker run --rm -p 8000:8000 gcr.io/stately-math-273218/my-python-app test

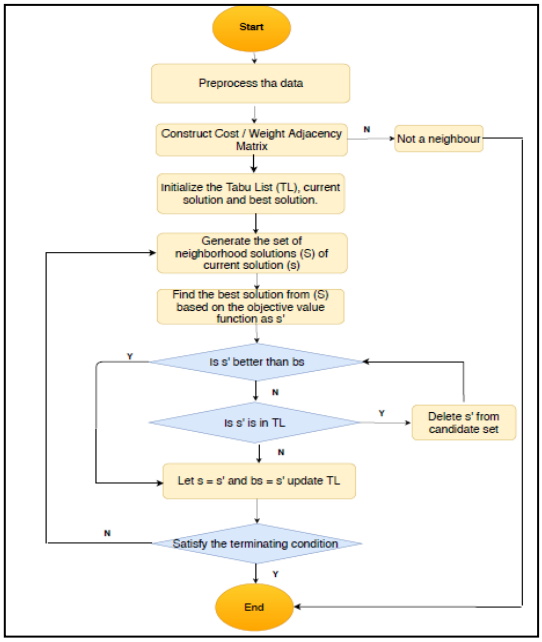

Fig. 6. Flow of implementation of Tabu search in solving routing problem in vehicular networks.

5. Experimental Results

Two types of evaluation are done to evaluate the proposed services in the framework. Applications are developed using Python language and deployed as Docker images in a dedicated web server. The services are accessed from different client machines using different operating systems and web browsers. This evaluation measures the efficiency of the microservices. In order to evaluate the performance of the suggested trust estimation and security modules, a simulation environment is established using OMNET++, Sumo and Veins tools. We used OMNET++, to implement our proposed TM framework. The SUMO, traffic simulator is used to create traffic flows, and the Veins system is used to model the vehicular climate. To ensure scalability, different traffic flow rates were used. For an urban road network representation, the Al Ain city map is chosen. A SUMO NetEdit map for the selected region is illustrated in Figure 7. We divide the road network into equal squared areas and position an RSU in the center of each area to model the proposed trust model, as shown in the figure. Each RSU calculates the traffic condition of the road links it supervises and disseminates it to vehicle agents within its coverage at the end of a predefined time interval.

Fig 7. Al-Ain City Map

Table 3. Environment setup for accessing cloud services

Hardware	
Processor	Intel R Core (TM) i7-7600
RAM	8.00 GB
CPU Clock Speed	2.80 GHz
Software and Simulation set up	
Operating System	Windows 10 64 bit
Cloud Environment	Google Cloud
Application Development	Python, OMNET++
Networking monitoring tool	PRTG Network monitor.
Web Server	Apache (2.4.46)

The traffic condition information is then used by vehicle agents to monitor the dissemination of reliable information in a multi-hop fashion through V2V communication. We use the IEEE 802.11p implementation available in Veins for the physical and MAC layers.

Table 3, illustrates the various hardware, software and simulation requirements needed to implement the services in cloud environment. Various metrics like packet delivery ratio, accessing time, time taken to build Docker images and CPU utilization time are used to evaluate the cross platform performance of the services. In addition, the services are tested individually for their performances. PRTG networking monitoring tool is used to monitor various networking activities of services.

Table 4. *Environment setup for simulation*

Physical layer	Frequency band	5.9 GHz
	Bandwidth	10 MHz

Service Info	Processor	CPU (GHz)	RAM (GB	Operating System	Browser	Response Time (s)	Time taken to build Docker images (s)	Bandwidth occupied (GHz)	Packet loss ratio (%)	CPU utilization time (s)
Trust Evaluation	Intel R Core i7 -7600	2.80	8.00	Windows	Chrome	0.23	0.35	1.12	2	0.22
				Linux	Firefox	0.24	0.37	1.26	2	0.24
				Ubuntu	Midori	0.24	0.39	1.34	3	0.26
Routing	Intel R Core TM i5 6200	2.30	4.00	Windows	Chrome	0.25	0.40	1.42	2	0.28
				Linux	Firefox	0.26	0.41	1.45	3	0.32
				Ubuntu	Midori	0.25	0.42	1.52	3	0.36
Message dissemination	Intel R Core i5-4200U	2.6	4.00	Windows	Chrome	0.26	0.50	1.61	2	0.32
				Linux	Firefox	0.28	0.53	1.67	3	0.36
				Ubuntu	Midori	0.32	0.55	1.72	2	0.41

	Radio range	~ 360 meter
MAC layer	MAC bit rate	6 Mbps
	Mac delay	20 millisecond
	Data frequency	5 Hz
Networking and application	Urban area	Al Ain city
	Area of interest	6500 x 4000 meter
	Maximum speed	100
	Message size	100 byte
	Number of messages	10
	Simulation time	900 s
	Number of runs	5
	Confidence level	95%
Scenario 1	% of Malicious vehicles	{10, 15, 20, 25, 30, 35, 40}
Scenario 2	Traffic flow rates	{2000, 4000, 6000, 8000} Vehicle/hr.

Similarly, Table 4, illustrates the parameter details of simulation environment to evaluate the performance of the proposed TMS.

5.1. Overall performance of the suggested services

The first section illustrates and explains various performance analysis of the suggested services. Table 5 illustrates the overall performance of the suggested trust evaluation service. From the figures, the suggested trust evaluation service performs well in Chrome browser, implemented in Windows operating system. Table 6, illustrates the efficiency of the trust model in detecting the malicious nodes in the vehicular environment by gradually increasing the number of malicious nodes in the simulation, and Table 7 depicts the overall performance of the suggested routing service in simulated environments. From the results, it is evident that the proposed services perform better in cross platform platforms and simulated environments.

Table 6. Performance of suggested Trust evaluation service

No of nodes	No of malicious nodes	No of nodes detected	Time taken to detect (s)	Accuracy (%)
10	5	5	0.124	100
20	7	7	0.143	100
30	14	14	0.163	100
40	22	19	0.227	86.36
50	32	30	0.242	93.75
60	44	40	0.265	90.90
70	56	52	0.327	92.85
80	62	59	0.356	95.16
90	72	68	0.443	94.44
100	80	75	0.521	93.75

Table 5. Overall performance of the proposed microservices in cross platform environments

Table 7. Performance of suggested routing service

No of Nodes	Optimal Cost of Tabu Search	No of iterations taken by Tabu search
10	312	10
20	350	15
30	411	22
40	467	34
50	510	41
60	578	53
70	684	62
80	768	74
90	831	82
100	945	91

5.2. Comparison with other referred studies

In the second experimental analysis, the performance of proposed trust estimation service and routing services are compared with other related referred studies. In the first comparison, the execution time of the referred [25] and the proposed routing algorithms to calculate the optimal cost to find the best route is analyzed. Figure 8, illustrates the results which depicts the better performance of the suggested routing mechanism over the referred routing model. The proposed routing algorithm consumes less number of resources thus minimizing the CPU utilization time over the proposed study. In the second comparison, the proposed trust evaluating service is compared with the performance of other reputed trust models, which operates in a similar fashion of the proposed trust model. The run time storage consumed by the models are analyzed. Since the proposed model uses simple concepts to evaluate the trust, it consumes less run time storage over other trust evaluation models. From the results portrayed in Figure (9), it is evident, that the proposed trust model performs better over the referred trust models.

6. Future Enhancements

The study can be further extended by implementing various applications like autonomous billing services, parking services and platooning services as micro services. The experiments are conducted using a dedicated server connected to multiple clients. In future, the services has to be deployed in distributed cloud servers

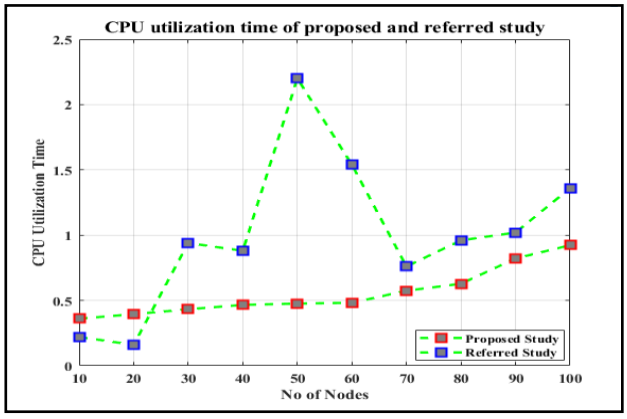

Fig. 8. Execution time comparison between the proposed and referred study [25]

from which the clients working in different platforms are allowed to access and integrate the suggested services in their applications. The services must be accesses from a public Docker hub hosted in cloud servers.

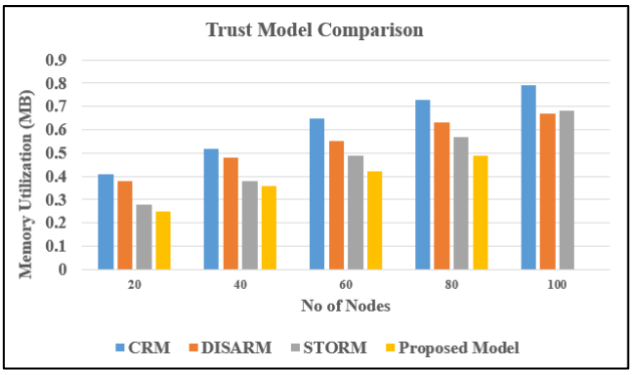

Figure 9: Performance of the proposed trust service with referred study [24]

7. Conclusion

This study has proposed a versatile framework to assimilate three mandatory micro services like trust evaluation, reliable message dissemination among vehicular entities and routing services for autonomous vehicles. Further, the study has proposed a versatile CipherText based encryption/decryption algorithm to ensure authenticated and safe message dissemination among the vehicular entities. The main objective of this study is to facilitate the autonomous car manufacturers and autonomous solution developers to use these services in their existing vehicular solutions, minimizing their development time and cost effectively. Further, these services are planned to be implemented and executed in cross platform environments. To achieve this, advanced features of Docker and Containers are used to design, build and deploy the services as Docker images, whose instances can be integrated in the containers deployed in the client's local machine. Google Cloud platform is used to implement the features of Dockers to build images. The images created can be executed in the same local machine where they are deployed or subsequently implemented in different cross-platform environments and accessed via different browsers. Experimental results prove that the time taken to deploy and access the proposed services are comparatively low in different system environments. Many evaluations and comparisons are done with other referred studies to prove the efficiency and accuracy of the proposed services. Future enhancements related to the study is also highlighted in this paper.

Acknowledgement

This paper was supported by Emirates Center for Mobility Research of the United Arab Emirates University (grant 31R271).

References

[1] Altche, F., Qian, X., & De La Fortelle, A. (2017). An algorithm for supervised driving of cooperative semi-autonomous vehicles. IEEE Transactions on Intelligent Transportation Systems, 18, 3527–3539.

[2] Anderson, J. M., Nidhi, K., Stanley, K. D., Sorensen, P., Samaras, C., & Oluwatola, O. A. (2014). Autonomous vehicle technology. Santa Monica, CA: Rand Corp.

[3] Arbolino, R., Carlucci, F., Cira, A., Ioppolo, G., & Yigitcanlar, T. (2017). Efficiency of the EU regulation on greenhouse gas emissions in Italy. Ecological Indicators, 81, 115–123.

[4] Arnaout, G., & Arnaout, J. P. (2014). Exploring the effects of cooperative adaptive cruise control on highway traffic flow using microscopic traffic simulation. Transportation Planning and Technology, 37, 186–199.

[5] Bagloee, S. A., Tavana, M., Asadi, M., & Oliver, T. (2016). Autonomous vehicles. Journal of Modern Transportation, 24, 284–303.

[6] El Sayed, H., Zeadally, S., & Puthal, D. (2020). Design and evaluation of a novel hierarchical trust assessment approach for vehicular networks. Vehicular Communications, 24, 100227.

[7] Baldini G, Fröhlich P, Gelenbe E, Hernandez-Ramos JL, Nowak M, Nowak S, Papadopoulos S, Drosou A, Tzovaras D. IoT Network Risk Assessment and Mitigation: The SerIoT Approach.

[8] Guleng S, Wu C, Chen X, Wang X, Yoshinaga T, Ji Y. Decentralized trust evaluation in vehicular Internet of Things. IEEE Access. 2019 Jan 17; 7:15980-8.

[9] Lotz J, Vogelsang A, Benderius O, Berger C. Microservice Architectures for Advanced Driver Assistance Systems: A Case-Study. In2019 IEEE International Conference on Software Architecture Companion (ICSA-C) 2019 Mar 25 (pp. 45-52). IEEE.

[10] Lu D, Huang D, Walenstein A, Medhi D. A secure microservice framework for iot. In2017 IEEE Symposium on Service-Oriented System Engineering (SOSE) 2017 Apr 6 (pp. 9-18). IEEE.

[11] Alam M, Rufino J, Ferreira J, Ahmed SH, Shah N, Chen Y. Orchestration of microservices for iot using docker and edge computing. IEEE Communications Magazine. 2018 Sep 17;56(9):118-23.

[12] Qu Q, Xu R, Nikouei SY, Chen Y. An Experimental Study on Microservices based Edge Computing Platforms. arXiv preprint arXiv:2004.02372. 2020 Apr 6.

[13] Gopalasingham A, Herculea DG, Chen CS, Roullet L. Virtualization of radio access network by Virtual Machine and Docker: Practice and performance analysis. In2017 IFIP/IEEE Symposium on Integrated Network and Service Management (IM) 2017 May 8 (pp. 680-685). IEEE.

[14] Ramírez López M, Spillner J. Towards quantifiable boundaries for elastic horizontal scaling of microservices. In6th International Workshop on Clouds and (eScience) Applications Management (CloudAM 2017), Austin TX, USA, 5-8 December 2017, ACM.

[15] D. Namiot and M. Sneps-Sneppe, "On iot programming," International Journal of Open Information Technologies, vol. 2, no. 10, 2014. [5] C. Anderson, "Docker [Software engineering]," IEEE Software, vol. 32, no. 3, pp. 102–c3, 2015.

[16] W. Felter, A. Ferreira, R. Rajamony, and J. Rubio, "An Updated Performance Comparison of Virtual Machines and Linux Containers," Technology, vol. 25482, pp. 171–172, 2014.

[17] D. Liu and L. Zhao, "The research and implementation of cloud computing platform based on docker," 2014 11th International Computer Conference on Wavelet Actiev Media Technology and Information Processing (ICCWAMTIP), pp. 475–478, 2014

[18] B. I. Ismail, E. Mostajeran Goortani, M. B. Ab Karim, W. Ming Tat, S. Setapa, J. Y. Luke, and O. Hong Hoe, "Evaluation of Docker as Edge computing platform," In: 2015 *IEEE Conference on Open Systems (ICOS)*, pp. 130–135, 2015.

[19] Soleymani SA, Abdullah AH, Hassan WH, Anisi MH, Goudarzi S, Baee MA, Mandala S. Trust management in vehicular ad hoc network: a systematic review. EURASIP Journal on Wireless Communications and Networking. 2017 May 23; 2017(1):146.

[20] Chaker Abdelaziz Kerrache et al, "RITA: Risk-aware Trust-based Architecture for collaborative muti-hop vehicular communications", Security and Communications, Volume 9, pp 4428-4442, 2016

[21] Purba AP, Siswanto N, Rusdiansyah A. Routing and scheduling employee transportation using tabu search. InAIP Conference

Proceedings 2020 Apr 13 (Vol. 2217, No. 1, p. 030143). AIP Publishing LLC.

[22] O. Senouci, Z. Aliouat, S. Harous, "A Review of Routing Protocols in Internet of Vehicles and their Challenges", Sensor Review, Emerald Publishing, Vol. 39, No. 1, pp. 58-70, 2019.

[23] Ghosh, T., and Mitra, S. (2012, November). Congestion control by dynamic sharing of bandwidth among vehicles in VANET. In Intelligent Systems Design and Applications (ISDA), 2012 12th International Conference on (pp. 291-296). IEEE.

[24] Kravari K, Bassiliades N. StoRM: A social agent-based trust model for the internet of things adopting microservice architecture. Simulation Modelling Practice and Theory. 2019 Jul 1;94:286-302.

[25] Jia H, Li Y, Dong B, Ya H. An improved tabu search approach to vehicle routing problem. Procedia-Social and Behavioural Sciences. 2013 Nov 6; 96:1208-17.

[26] Förster D, Kargl F, Löhr H. PUCA: A pseudonym scheme with user-controlled anonymity for vehicular ad-hoc networks (VANET). In2014 IEEE Vehicular Networking Conference (VNC) 2014 Dec 3 (pp. 25-32). IEEE.

[27] Prema NK. Efficient secure aggregation in VANETs using fully homomorphic encryptions (FHE). Mobile Networks and Applications. 2019 Apr;24(2):434-42.

[28] Deng X, Xin X, Gao T. A location privacy protection scheme based on random encryption period for VSNs. Journal of Ambient Intelligence and Humanized Computing. 2020 Mar; 11(3):1351-9.

[29] Duan X, Liu Y, Wang X. SDN enabled 5G-VANET: Adaptive vehicle clustering and beamformed transmission for aggregated traffic. IEEE Communications Magazine. 2017 Jul 14;55(7):120-7.

[30] Truong NB, Lee GM, Ghamri-Doudane Y. Software defined networking-based vehicular adhoc network with fog computing. In2015 IFIP/IEEE International Symposium on Integrated Network Management (IM) 2015 May 11 (pp. 1202-1207). IEEE.

[31] Han Y, Tao X, Zhang X, Jia S. A clustered vnf-chaining scheme with delay guarantees in nfv-based vanets. In2018 IEEE International Conference on Communications Workshops (ICC Workshops) 2018 May 20 (pp. 1-6). IEEE.

[32] Djellali C, Adda M. An Enhanced Deep Learning Model to Network Attack Detection, by using Parameter Tuning, Hidden Markov Model and Neural Network. International Journal of Ubiquitous Systems and Pervasive Networks (JUSPN). 2021;15(01):35-41.

[33] Feltus C. AI'S Contribution to Ubiquitous Systems and Pervasive Networks Security-Reinforcement Learning vs Recurrent Networks. International Journal of Ubiquitous Systems and Pervasive Networks (JUSPN). 2021;15(02):1-9.