Ad Hoc Vehicular Fog Enabling Cooperative Low-Latency Intrusion Detection

Azzam Mourad, *Senior Member, IEEE*, Hanine Tout, Omar Abdel Wahab, Hadi Otrok, *Senior Member, IEEE*, and Toufic Dbouk

***Abstract*—Internet of Vehicles and vehicular networks have been compelling targets for malicious security attacks where several intrusion detection solutions have been proposed for protecting them. Nonetheless, their main problem lies in their heavy computation, which makes them unsuitable for next-generation artificial intelligence-powered self-driving vehicles whose computational power needs to be primarily reserved for real-time driving decisions. To address this challenge, several approaches have been lately presented to take advantage of the cloud computing for offloading intrusion detection tasks to central cloud servers, thus reducing storage and processing costs on vehicles. However, centralized cloud computing entails high latency on intrusion detection related data transmission and plays against its adoption in delay-critical intelligent applications. In this context, this article proposes a vehicular-edge computing (VEC) fog-enabled scheme allowing offloading intrusion detection tasks to federated vehicle nodes located within nearby formed *ad hoc* vehicular fog to be cooperatively executed with minimal latency. The problem has been formulated as a multiobjective optimization model and solved using a genetic algorithm maximizing offloading survivability in the presence of high mobility and minimizing computation execution time and energy consumption. Experiments performed on resource-constrained devices within actual *ad hoc* fog environment illustrate that our solution significantly reduces the execution time of the detection process while maximizing the offloading survivability under different real-life scenarios.**

***Index Terms*—*Ad hoc* vehicular fog, cooperative intrusion detection, federated vehicles, Internet of Things (IoT), Internet of Vehicles (IoV), mobile-edge computing (MEC), multiobjective optimization, security, offloading, resource management, vehicular-edge computing (VEC), vehicular fog federation.**

I. Introduction

INTERNET of Vehicles (IoV), an essential paradigm of the Internet of Things (IoT) [1] related to intelligent transportation systems (ITS), are networks of interconnected vehicles and roadside units (RSUs) that exchange information to detect, prevent, and manage traffic problems [2]. For example, vehicles can exchange information about road conditions in order to avoid traffic jams. Due to their expansion, several commercial applications (e.g., multimedia and infotainment) have also been integrated into such networks for marketing and business purposes. Unfortunately, vehicular networks have been an appealing target for countless malicious attacks (e.g., impersonation and bogus information dissemination) due to their infrastructureless and distributed nature [3]. Such attacks are likely to result in catastrophic consequences ranging from loss of lives (in case of traffic management applications) to loss of revenue (in case of commercial applications) [4]. This demands setting up strict security measures (i.e., intrusion detection and prevention systems) to protect the vehicular network from being an easy target for attackers. This is however a difficult task in such networks that have special restrictions and needs. The main restriction is related to the massive computation load that is often needed for analyzing big intrusion detection data. For example, previous results showed that running an intrusion detection system (IDS) over a data set of 10-MB size can consume up to 212 J of energy, up to 100 MB of RAM and up to 460 s of CPU on a mobile device, with an execution time of 459 s [5]. On the other hand, with the rapid evolution in ITS and the integration of artificial intelligence (AI) capabilities into today's smart devices (i.e., autonomous vehicles), the resources that are available on smart vehicles have to be efficiently exploited to ensure the success of such technology and avoid undesirable consequences. In other words, the resources that are available on AI-powered vehicles should be primarily dedicated to making real-time driving decisions (e.g., detecting pedestrians, cyclists, etc.), rather than to security assurance. Therefore, there should be a smart load distribution mechanism to efficiently decide where the intrusion detection tasks should be executed. Motivated by this idea, we design in this work a distributed multilayer offloading approach that takes advantage of the emerging edge computing technology to perform intrusion detection in vehicular communication systems in an efficient and resource-aware manner.

A. Problem Statement

Several approaches [6]–[10] have proposed to take advantage of the cloud computing technology to deal with the resource restrictions on smart devices (e.g., vehicles and

Manuscript received April 13, 2020; revised June 17, 2020; accepted June 30, 2020. Date of publication July 10, 2020; date of current version January 7, 2021. This work was supported in part by the Lebanese American University, in part by Universite du Quebec en Outaouais, and in part by Khalifa University. *(Corresponding author: Azzam Mourad.)*

Azzam Mourad and Hanine Tout are with the Department of Computer Science and Mathematics, Lebanese American University, Beirut 961, Lebanon (e-mail: azzam.mourad@lau.edu.lb; hanintout@hotmail.com).

Omar Abdel Wahab is with the Department of Computer Science and Engineering, Université du Quebec en Outaouais, Gatineau, QC 8Y 3G5, Canada (e-mail: omar.abdulwahab@uqo.ca).

Hadi Otrok is with the Center of Cyber-Physical Systems, Department of EECS, Khalifa University, Abu Dhabi, UAE (e-mail: hadi.otrok@ku.ac.ae).

Toufic Dbouk is with Samsung Electronics America, Ridgefield Park, NJ 07660 USA (e-mail: toufic.dbouk@gmail.com).

Digital Object Identifier 10.1109/JIOT.2020.3008488

mobiles), using the concept of offloading. Offloading consists of migrating intensive computations from a resource-constrained device to some external platforms (i.e., cloud data centers) that are characterized by their massive computing and storage capacities. However, the performance of this approach is hindered by the centralized architecture of cloud computing, which entails long latency for data transmission from/to cloud data centers. This makes it difficult to keep up with the Quality-of-Service (QoS) requirements of the delay-sensitive applications, such as healthcare management, connected vehicles, and video streaming. These challenges had led to the rise of a new architecture called mobile-edge computing (MEC) [11] whose main idea is to deploy computing and storage resources at the edge of the network, thus considerably reducing the latency of computing services. Vehicles promise to contribute in boosting the rebound of edge intelligence, thanks to the unprecedented proliferation in the number of connected vehicles and amounts of computing and storage resources available on them. This pushed the research community to design several solutions [12]–[14] to address the problem of offloading in MEC cloud-enabled vehicular networks. These contributions seek mainly to allow users (i.e., vehicles and pedestrians) to offload their computation tasks to other vehicles, that might exploit their excess of computing resources to serve them, or to fog resources placed at the edge [15]–[17]. Despite their importance, to the best of our knowledge, none of these approaches has yet considered the offloading of security services in the MEC fog-based vehicular environment within IoT and IoV. To fill in this gap, we propose in this work *ad hoc* fog within vehicular federation approach for efficient intrusion detection which generates optimal offloading decisions that minimizes the energy consumption and execution time and maximize the clustered fog survivability, while considering the mobility and available resources of the vehicles in the design of the solution. The second limitation of the existing approaches is that they demand Internet connection and preset infrastructure to carry out the offloading process. Unlike these approaches, our solution takes advantage of the Wi-Fi direct technology to alleviate these burdens and reduce the offloading cost for users within the infrastructureless environment.

B. Contributions

The propositions in this work target security services in general, and can be extended to more general problems requiring critical computation and low-latency tasks. In this work, we focus on IDSs due to their complexity and the computation overhead that they put on vehicles. We then propose a novel vehicular-edge computing (VEC) fog-enabled scheme within IoV formed among nearby federated vehicles while providing sustainable and cooperative intrusion-detection service through intelligent and efficient multilayer computation offloading. Our approach consists of three main components within the federation of vehicles: 1) master node; 2) requesting node; and 3) serving node. The master node is considered to be the owner of the fog within a specific Wi-Fi direct group and is hence responsible for carrying out the communications among the different nodes within its cluster. Once the requesting node sends an offloading request to the master, the latter runs the vehicular intelligent offloading distributor (VIOD) module to decide on whether the security scanning task should be done locally or in a remote fashion (i.e., offloaded). To come up with such a decision, the VIOD analyzes several parameters gathered from the requesting node and the other vehicles in the cluster using *profiler* and *Fog Observer* agents. Specifically, the *profiler* agent collects parameters, such as size of data to be offloaded, number of nodes that exist in the underlying vehicular fog, status of each node (i.e., idle and moderate critical), battery level of the vehicle, and CPU and energy consumption on each of these vehicles.

On the other hand, the *Fog Observer* is in charge of collecting mobility-related data from vehicles, such as geo-coordinates, speed, and distance. More specifically, this agent periodically computes each vehicle's distance with respect to the master node using the vehicle's geo-coordinates [obtained using the global positioning system (GPS)] to determine the time needed for the vehicle to get out of the master's network range [i.e., out-of-range (ORT) metric]. This enables the master to compare the ORT of each vehicle with the expected execution time of the underlying task to rule out those vehicles that are likely to leave the cluster's range area before completing the task. This step is of prime importance since it contributes in both: 1) reducing the overhead of the offloading algorithm by excluding the set of ineligible vehicles and 2) improving the overall QoS of the intrusion detection process through guaranteeing that task executors will not leave the network prior to finishing the execution of their tasks.

In fact, vehicles often change their speed based on road conditions, such as driving in cities, on highways, or even from one street to another. This variation affects the structure of the *ad hoc* fog as some vehicles participating in the federation might get out of their cluster's communication range. To better illustrate this idea, consider a vehicle driving with high speed at the time of taking an offloading decision. This vehicle is selected to perform a part of the intrusion detection task and is expected to complete its assigned task. However, it might not be able to complete its task since it might get out of the master node's range before finishing execution. As a result, the efficiency and reliability of our framework is compromised. In such cases, the framework should be able to decide on offloading to nodes that are guaranteed to completely execute their tasks within the time limit to meet with the needed QoS level. In another scenario, assume that a node requested to offload in the VEC network and that serving nodes are ready to utilize their resources to carry out the intrusion detection task. However, by the time the serving nodes finish the execution of their assigned tasks, the requesting node had already exited the master node's range. As a result, the VEC *ad hoc* fog resources were consumed without gaining any benefits. In such cases, the offloading decision should be denied saving and avoiding waste of resources. Our proposed framework uses vehicles' speed and distance among them to calculate the time needed for each node to get out of the master node's communication range. It also finds the maximum number of chunks that a node can serve before it exits the vehicular fog. More specifically,

the framework focuses on the offloading survivability of the fog. The survivability aspect plays an important role in determining whether the underlying vehicular fog guarantees the execution of the offloading decisions taken. We consider the offloading survivability of the vehicular fog as its ability to offload all tasks to the serving nodes and sends the result back to the requester node before any of the involved nodes gets out of the cluster's range and at the same time, before the requesting node exits the network's range.

By analyzing all these collected parameters, the VIOD then decides on whether offloading the intrusion detection task would improve the overall performance of the process or not. If an offloading decision is adopted, the VIOD computes, using the NSGAII genetic algorithm [18], the appropriate distribution of the intrusion detection data over the set of eligible federated vehicles in such a way to minimize energy and execution time, and maximize ORT. In summary, the main contributions of this work are listed as follows.

1) Elaborating a novel *ad hoc* vehicular fog scheme composed of federated vehicles enabling cooperative security and intrusion detection services that capitalize on the Wi-Fi direct technology to provide vehicles with an Internet-independent offloading strategy. To the best of our knowledge, this work is the first that provides intrusion-detection service in an *ad hoc* fog within vehicular federation built from participating vehicles, while eliminating the need to rely on Internet connection or any other preset infrastructure. Our solution is designed in such a way to be independent from the availability and quality of the Internet connection, without entailing any additional charges for users.
2) Formulating the intrusion detection task offloading problem as a multiobjective optimization model which takes into consideration performance improvement and maximization of offloading survivability time. The optimization problem is solved using a genetic algorithm that relies on statistical mobility and resource data collected from vehicles to generate offloading solutions that minimize execution time and improve the QoS of the cooperative distributed intrusion detection process. Moreover, our experiments were performed on resource-constrained devices reflecting actual *ad hoc* vehicular fog federation while considering real-life factors, such as high-mobility, survivability of the formed cluster of vehicles, and different scenarios of available resources.

C. Organization

The remainder of this article is organized as follows. In Sections II and III, we present an architectural overview of our solution and explain the different components and modules. In Sections IV and V, we formulate the distributed offloading problem as a multiobjective optimization model and propose an NSGAII genetic approach to solve it in an efficient manner. In Section VI, we provide thorough experimental evaluations of our solution under different scenarios and parameters. In Section VII, we provide a security analysis and highlight some challenges for potential solutions. In Section VIII, we survey the related work and highlight the originality of our solution. Finally, in Section IX, we conclude this article and summarize the main insights driven from our work.

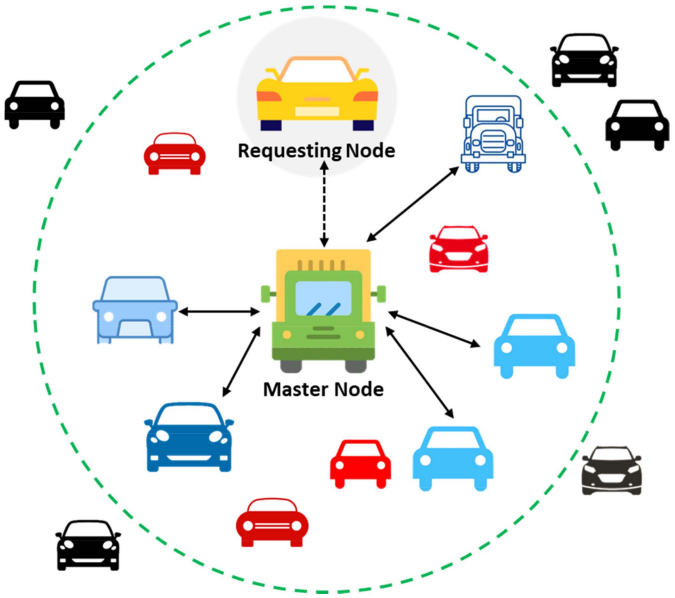

Fig. 1. *Ad hoc* vehicular fog architecture composed of federated vehicles.

II. Scheme Overview

This section describes, in a nutshell, our proposition illustrated in Fig. 1. The proposed approach is capable of creating an *ad hoc* vehicular fog and intelligently offloading requests between nodes in a vehicular federation. In the figure, we differentiate between five types of vehicles. The yellow one resembles a *Requesting Node*, the green one resembles the *Master Node*, and blue ones resemble *Serving Nodes*. Red vehicles resemble *Serving Nodes* within the *ad hoc* vehicular fog, which are excluded from the intelligent offloading decision. Therefore, even though they are part of the fog federation and close to the *Master Node*, they are being discarded by the Intelligent Offloading Distributor module since they are not able to satisfy the vehicular constraints. One reason behind excluding them could be their relatively high moving speed compared to other nodes which leads to going out of the network range before finishing executing their assigned tasks. This critical criterion is made available to the *Master Node* via the Fog Observer module. Finally, black vehicles show vehicles that are outside of the *ad hoc* vehicular fog federation and not being used in the offloading decision.

Our approach creates an *ad hoc* vehicular fog through Wi-Fi direct in order to build a dynamic environment of federated vehicles for sharing resources, creating cooperation opportunities, and offloading between different nodes in order to provide efficient and reliable IDS. While different nodes are involved in each offloading process, each of which runs several modules that differ based on the node type. The communication between a requester and serving nodes is established through the Master, which is considered in this work as the group owner (GO) of the created *ad hoc* vehicular fog. A secondary Master node is also considered in the proposed architecture in

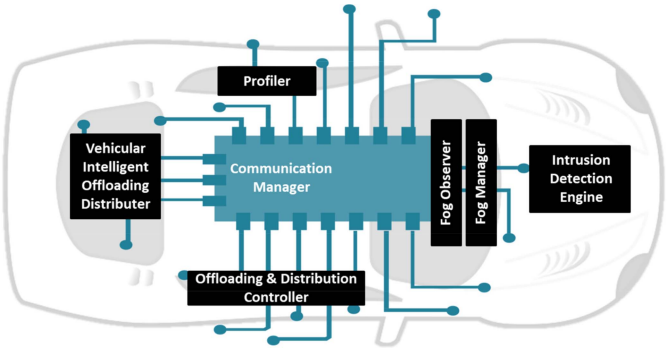

Fig. 2. Vehicular node modules.

order to replace the main one in case failure. A Requesting Node is a node in the *ad hoc* vehicular fog that requests offloading the execution of a task. It is the responsibility of the Master node then to evaluate whether offloading augments the performance of the requesting node or not. In the latter case, the requesting node executes the task locally. An *ad hoc* vehicular fog manager, profiler, communication manager, and intrusion detection engine, are all modules that run inside a node in order to handle the communications, offload the tasks, and execute the detection process. A Master Node is the GO of a particular Wi-Fi direct group, responsible of managing the *ad hoc* vehicular fog federation as well as handling the offloading requests. Based on several metrics, which are further explained in this article, the Master finds the nodes which are more suitable to handle the offloading requests. Additionally, it communicates data between the nodes in the *ad hoc* vehicular fog. Besides the modules of a requesting node, the Master includes a VIOD, Offloading and Distribution Controller and a Fog Observer, which are all detailed in the following section. Finally, a Serving Node is a surrogate node in the *ad hoc* vehicular fog that answers an offloading request to the Master node and on behalf of the Requesting one. The serving node performs the required task and communicates the results back to the master node. It runs the same modules as a requesting node.

III. *Ad Hoc* Vehicular Fog

This section details each module of the federated vehicles involved in the *ad hoc* vehicular fog as illustrated in Fig. 2, in addition to the interaction among them. An algorithm for the Election of Master Nodes is proposed, which is inspired by our previous work published in [19]. The main component of this algorithm is a multicriteria QoS function described as follows:

$$\text{QoS}(i) = w1 \times BW(i)w2 \times N(i)w3 \times \text{DistRatio}(i)/w4 \times \text{VelRatio}(i). \quad (1)$$

This function consists of several metrics, such as bandwidth, connectivity, velocity, and residual distance. The bandwidth is considered to ensure the reliability of the vehicular network, the connectivity is considered to increase the coverage of the elected master nodes, while the velocity and residual distance are considered to maintain the stability of the network and minimize the chances of link failures caused by the elected master nodes. In our algorithm, vehicles periodically broadcast *HELLO* messages [19], which contain information on their QoS metrics (i.e., bandwidth, connectivity, velocity, and residual distance). Based on this information, each vehicle participates in two election processes, one for a primary master node and one for a secondary master node. To select the primary master node, vehicles compute the above-described QoS function while giving equal weights for the different metrics (i.e., w1=w2=w3=w4). Each vehicle then votes for the vehicle that enjoys the local maximal QoS value. To select the secondary master node, vehicles compute the QoS function while this time giving more weights for the mobility metrics (i.e., velocity and residual distance). The idea is to increase the probability of having the secondary master node longer in the underlying cluster in case the primary head disconnects. Again, each vehicle votes for the vehicle that enjoys the local maximal QoS value. In case the same vehicle enjoys the maximal QoS value in both election processes, then nodes vote for the vehicle that has the second local maximal QoS to be the secondary master. A vehicle is also entitled to vote for itself, provided that it has the local maximal QoS value. The vehicles employ special *HELLO* messages, called Election messages, to locally broadcast their votes. Once the elections are completed, the primary and secondary elected vehicles acknowledge to serve as a master nodes through forwarding an *Ack* message. A cheating prevention mechanism, which motivates vehicles to reveal their true QoS metrics is also described in [19].

Based on the type of the node on which the fog manager is running, it will act from different perspective. When running on the Master node, this module forms and monitors a vehicular fog within a certain range. It handles requests from nodes willing to join the fog and keeps track of each connected node status. The profile of each node is communicated with the fog manager, dynamically and frequently updated, in order to serve in the offloading decision process. Following a discovery phase in which the manager looks for a GO/Master node to join the cluster, a connection is established with the latter to communicate the nodes profiles (created by the Profiler module) with the fog manager. The latter further takes the responsibility of handling the connection (reconnect/disconnect) in case of failures. The Profiler in this work is a service running as a separate process that uses a native code to access the kernel's information center and capture runtime system information of a node. It collects energy consumption and monitors execution time. Such data get updated regularly and can be requested on-demand basis. Based on the gathered information, the profiler classifies each node in the *ad hoc* vehicular fog as *Idle, Moderate*, or *Critical*. The Communication Manager is a data socket layer that identifies a set of protocols defining and abstracting the communication based on each node's requirements. According to the node's type and the operation to be performed, the communication manager arbitrates either a P2P or a client–server communication mechanism. Furthermore, it communicates traces between the requester and *ad hoc* fog manager module and the offloading distribution controller of the master node.

The intrusion detection engine can be developed either based on signatures that are matched with a predefined malicious behavior data set or through anomaly detection. In the former case, a variety of algorithms can be used; like Aho–Corasick, Low Memory Keyword Trie, and Wu–Manber, which are currently implemented by snort pattern matcher [20]. As for the latter case, genetic algorithms and data mining techniques can be used to build such engines which takes a baseline of normal behavior and detect deviations, notifying when a threshold is crossed. The proposed architecture is applicable for any of these IDSs. Yet, in this specific work, a signature-based intrusion detection engine is built on top of the Aho–Corasick algorithm, which matches system calls gathered in real time against a predefined data set. To accelerate the traversals and searching process, finite state machine (FSM) of transitions and failure links between internal nodes is leveraged. Working on a predefined data set, the algorithm moves linearly with respect to the input length and number of matched patterns. The constructed FSM is used for pattern matching in a background thread running on the participating node. When a local execution is advised by the Intelligent Offloading Distributor, the detection engine scans data on the node in which it is running and takes action accordingly like raising alerts or blocking execution. However, in case an offloading decision is taken, the engine scans the data sent by the Requester node through the Master and sends the results back to the latter through the Communication Manager. Participating nodes are not tied by the same detection engine, but rather different engines/versions can run on each node to improve the detection rate.

The Fog Observer is a crucial module developed as a service and runs on the Master Node. It allows the Master Node to calculate and save its speed, in addition to keeping a continuous record of the time needed for each node to get out of its network range (ORT). In order to obtain this ORT factor, the module calculates various metrics for each node. The first metric is the node's geo-coordinates which are calculated via the underlying platform's GPS and network provider. The second metric is the distance between the node and the Master Node which is calculated via the obtained geo-coordinates. The third metric is the speed of the node which is calculated using the above two metrics. Finally, the Master Node obtains and saves the ORT for each node in the *ad hoc* vehicular fog. The data is updated periodically by the fog manager and other modules can request on-demand updates. The VIOD module is in charge of intelligently deciding whether offloading augments the performance of the Requesting Node or not. The distributor incorporates several criteria from the *ad hoc* vehicular fog in order to obtain a decision. The profiler provides the following information: 1) size of the data to be offloaded; 2) the number of cluster nodes; 3) status of each cluster node (idle and moderate critical); 4) battery level on each node; and 5) their CPU and energy consumption. Moreover, the module requests additional information for each node represented by its out-of-range time (ORT). The Fog Observer supplies the module with the time needed for each node to get out of the Master Node's network range. As such, it is capable of determining the ORT of the Requester Node and checking whether the node's assigned chunks are within the requester's ORT. It is also capable of excluding nodes that are unable to satisfy the Vehicular constraints. As a result, the module now has a complete view of the *ad hoc* vehicular fog enabling it to obtain a smart offloading decision in a vehicular environment. If an offloading decision is taken, the module then requires the Requester Node to send the data for distributing it based on the taken decision.

The Offloading and Distribution Controller module runs on Master nodes only. It is responsible for distributing the data on selected nodes in the vehicular fog beside gathering and communicating the results among them. It is implemented as a service that receives the decision of the Intelligent Offloading Distributor and disseminates it accordingly. It runs as a background thread for each Serving node dividing data into equal segments over available nodes in the formed vehicular fog. If equal segmentation is not feasible, the last segment is left to be larger than the rest. Each thread holds the responsibility of data transmission in cooperation with the Communication Manager and reliably to the node. For higher computation efficiency, parallelism is applied for data distribution. Detailed information regarding the scanning process duration, number of malicious items, and the items themselves are also communicated with the Distribution Service, which sends the result back to the Requester node through the Communication Manager. In case of failures in threads or nodes, the Master triggers the Requester node for recovery.

IV. Optimization Model and Problem Formulation

In this section, we explain our multiobjective optimization problem and prove that it is NP-hard by reduction to the multiobjective multidimensional Knapsack (MOMKP) problem. Existing intrusion detection solutions impose heavy computations on smart devices (i.e., autonomous vehicles), which makes them unsuitable for next-generation AI-powered vehicles as computational power should rather be primarily reserved for making real-time driving decisions. Hence, the need for an offloading-based solution where some scanning tasks are offloaded to be executed remotely. Consider a smart device $D(C, N)$ where C represents the size of data to be scanned by D, and N represents a set of connected nearby nodes forming an *ad hoc* vehicular fog such that $N_1+N_2+\cdots+N_j = N$. The offloaded data will be divided into chunks to be scanned on different nodes in order to release device D from intensive processing. Let C_i be a chunk of the data such that $C_1 + C_2 + \cdots + C_n = C$. Each C_i has its own independent demands with respect to execution time and energy consumed on computations. The demands are divided into two parts: 1) local and 2) remote. The local demands refer to the execution time and energy consumed for computations on the device D itself. The remote ones refer to the execution time and energy consumption on a remote device N_j such that $N_j \neq D$. C_i demands are dependent on the executing node N_j. On the other hand, the survivability of the vehicular fog is critical to guarantee the execution of the offloaded tasks. Determining the finest distribution of chunks, on the different node, which would lead to the best outcome with respect to all these constraints, is a challenging problem.

TABLE I
MODEL NOTATIONS

Variable	Description
$\|VC\|$	Number of nodes in the ad-hoc vehicular fog
x_{Nj}	Decision variable that define the percentage of data to be scanned on node j.
$Energy^{pr}_{Nj}$	Energy consumed on local processing.
$Energy^{tr}_{Nj}$	Energy consumed for chunks offloading.
$Energy^{idle}_{Nj}$	Energy consumed on Wi-Fi turned on and scanning.
$ExecutionTime^{pr}_{Nj}$	Execution time for local processing.
$ExecutionTime^{tr}_{Nj}$	Execution Time for chunks offloading.
$T^{survivability}_{Nj}$	Average time remaining after the offloading process finishes execution and before the Requesting Node gets out of range.
W_{F_E}	Weight for the energy objective function.
W_{F_T}	Weight of the execution time objective function.
W_{F_S}	Weight of the survivability factor.

Definition 1: Given some chunks of data that need to be analyzed on vehicle V, where each chunk C_i has local energy consumption $E_{c_i}^{L}$, local execution time $T_{c_i}^{L}$, remote energy consumption $E_{c_i}^{N_j}$, and remote execution time $T_{c_i}^{N_j}$, a network bandwidth B, latency L, and a set of vehicular nodes N in an *ad hoc* vehicular fog VC where each node N_j has an out-of-range time n_j^{ORT}, find the best distribution of these chunks over N such that the execution time and energy of the analysis on the vehicle V are minimized and the offloading survivability of VC is maximized.

This is a complex problem since it is heavily impacted by the number of possibilities these chunks can be distributed over participating nodes in VC, i.e., on which nearby vehicle each chunk will be scanned. This depends on the number of ways n different chunks can be allocated into m different sacks with $c1$ chunks in the first sack, $c2$ in the second sack, etc., such that $c_1 + c_2 + \cdots c_m = n$. In our model, the number of sacks m refers to the number of nodes in the *ad hoc* vehicular fog since each chunk can be executed on any vehicle. In order to highlight more on the complexity of the problem, consider an *ad hoc* vehicular fog consisting of ten nodes where each node is serving a chunk of the data to be scanned. Accordingly, each chunk can be served either locally or on any of the nine remaining nodes. Therefore, we have 10^{10} different possible distributions. This number can significantly and radically increase as the number of nodes and chunks increase reaching 10^{100} or even higher number if the number of nodes increases as well.

Theorem 1: The problem described in Definition 1 is NP-Hard.

Proof: The MOMKP [21] problem is reduced to our problem. The MOMKP is defined as follows: given n items $(b_1, b_2, \ldots, b_n)$ where each one has m weights $(w_1^i, w_2^i, \ldots, w_m^i)$ and p values $(c_1^i, c_2^i, \ldots, c_p^i)$, distribute the items into different sacks such that the total p values is maximized without exceeding the m knapsack capacities. Consequently, an instance of our problem is constructed as follows.

1) Having $C_1, C_2, \ldots, C_i$ as chunks to be scanned, which represent items in the MOMKP, and setup N nodes in the *ad hoc* vehicular fog forming the sacks.
2) Set the values for each chunk (item) C_i to be the object functions $f_{t_{c_i}}$ and $f_{e_{c_i}}$, which define the cost scanning chunks in terms of execution time and energy, respectively.
3) Set the weights for each chunk (item) C_i to be the demands in terms of execution time and energy when served either locally or remotely on node N_j.
4) Set the total weights of each node as $\tilde{T}_{\text{Threshold}}$ and N_r^{ORT}, which form thresholds for execution time and survivability, respectively in order to maintain good performance and guarantee completion of assigned scanning tasks.

The target of our problem is to minimize the total cost of chunks distributed on different nodes while maximizing the survivability rather than maximizing the value in MOMKP. However, the problem is still essentially the same. Following the above reduction, a solution to our problem can be used as a solution to the MOMKP one. Therefore, our problem is NP-Hard. ■

We further formulate our multiobjective optimization model by adapting it to the vehicular domain. The problem aims to speed up the scanning process by reducing its execution time, maximize the offloading survivability in the fog in order to achieve a reliable intelligent offloading in VANETs, and minimize the power consumed on data analysis on vehicle V. Table I defines all the notations used within the below model. We consider in this work a highway mobility topology, which implies that the vehicles are moving in the same direction and the speed variation among the vehicles is deemed not be high. One of the main functions in the formulated optimization model is the offloading survivability time factor. The latter allows the model to intelligently decide on the size of the data to be offloaded to each vehicle based on each node's ORT. Below we list the full three optimization objectives.

Objective 1 (Minimizing Execution Time): The total execution/computation time of a task [depicted in (2)] consists of the local execution time on the device ($\text{ExecutionTime}^{pr}_{Nj}$) along with the transmission round-trip time ($\text{ExecutionTime}^{tr}_{Nj}$). The round-trip time represents the time to send the chunk for processing and receive the result, including the network latency

$$F_T = \min \sum_{Nj=1}^{|VC|} x_{Nj}\left(\text{ExecutionTime}^{pr}_{Nj} + \text{ExecutionTime}^{tr}_{Nj}\right) \times W_{F_T}. \tag{2}$$

Objective 2 (Maximizing Offloading Survivability Time): The offloading survivability time [depicted in (3)] is the average remaining time after all *Serving Nodes* have finished execution, but before the *Requesting Node* goes out of the transmission range ($T^{\text{survivability}}_{Nj}$). Algorithm 2 illustrates the

process of computing the offloading survivability

$$F_S = \max \sum_{Nj=1}^{|VC|} x_{Nj}\left(T_{Nj}^{\text{survivability}}\right) \times W_{F_S}. \tag{3}$$

We shed the light on two crucial constraints when determining the offloading survivability illustrated in the next equations. If any of the two constraints is violated, the solution is invalidated and neglected:

1) $N_r^{\text{ORT}} \geq F_T$;
2) $N_j^{\text{MAX}} \geq x_{Nj}$.

The first equation involves the Requester's ORT which indicates the time by which a *Requesting Node* gets out of the *Master Node's* network area. We represent a node having an *ORT* time as N_j^{ORT}. To further stress this point, assume a *Requester Node* ($N_j^{\text{ORT}} = 15$) decides to offload in the *ad hoc* vehicular fog at time t_1, it follows that N_j should receive the result from the *Master Node* before $t_1 + 15$ s. In case the total execution time is more than 15 s, the solution is flagged and considered as a violation to the constraint.

The second equation involves the maximum number of chunks that a node N_j can accommodated before it gets out of range. This is important in order to guarantee that a *Serving Node* is capable of executing all its assigned chunks before it actually leaves the network. We represent a *Serving Node* having three maximum allowed chunks as $N_j^{\text{MAX}} = 3$. To further emphasize this, assume a *Serving Node* $N_j^{\text{MAX}} = 3$, then it follows that this node can serve at most three chunks. If the node is assigned more than three chunks, it is flagged and considered as a violation to the constraint.

Objective 3 (Minimizing Energy Consumption): While computational energy consumption is not an issue in the case of vehicles, we chose to include this parameter in the optimization problem to account for the cases wherein mobile devices located inside vehicles might be chosen to carry out the offloaded IDS tasks. This increases the flexibility of our solution and boosts its applicability in a wider range of practical scenarios. The weight of this parameter can then be adjusted according to the underlying scenario, where for example, if the offloading nodes are vehicles, the weight of minimizing the energy consumption can be set to 0. On the other hand, when offloading nodes are mobile devices inside vehicles, the weight of this function can be accordingly increased. Technically speaking, the overall energy consumption of serving a chunk [depicted in (4)] consists of both the energy consumed locally for serving the underlying task and energy consumed through offloading the task. The local energy is spent on CPU processing (Energy_{Nj}^{pr}) while the offloading cost stems from the energy spent on data transmission (Energy_{Nj}^{tr}), including having the Wi-Fi radio turned on

$$F_E = \min \sum_{Nj=1}^{|VC|} x_{Nj}\left(\text{Energy}_{Nj}^{pr} + \text{Energy}_{Nj}^{tr} + \text{Energy}_{Nj}^{\text{idle}}\right) \times W_{F_E}. \tag{4}$$

After declaring the minimization and maximization functions, the multiobjective optimization problem becomes

$$\begin{aligned} F = & \begin{bmatrix} F_T \\ F_S \\ F_E \end{bmatrix} \\ \text{s.t.} \quad & n \in \mathbf{N} \\ & 0 <= x_{Nj} <= 100 \\ & F_T \leq \tilde{T}_{\text{Threshold}} \\ & W_{F_T} + W_{F_S} + W_{F_E} = 1. \end{aligned}$$

The first constraint is to make sure that the number of vehicles within the *ad hoc* vehicular fog is a natural number. Next, the second constraint ensures that the decision variable is between 0 and 100. In other words, the value of this variable determines the percentage of chunks to be scanned on that node. A value of 0 means that the device N_j will not participate in the detection process. The third constraint sets a deadline for reporting the scanning results. All objective functions defined above are assigned certain weights which define how important/critical attaining these objectives would be depending on the device state. An example could be prioritizing the offloading survivability where the remaining time after offloading within the same *ad hoc* vehicular fog could be more significant than the execution time and energy. For instance, a cluster having frequent disconnections of its nodes would be better off prioritizing the survivability factor over energy or execution time. The total of these weights sums up to 1 as defined in the fourth constraint.

V. Heuristic-Based Solution

In the following, we discuss our vehicular intelligent and resource-aware offloading algorithms shown in Algorithms 1–3. The input parameters of Algorithm 1 are the size of the population N, individual length L, crossover rate μ_c, and mutation rate μ_m as shown in line 1. Line 2 shows the final output of the algorithm, which is a set containing the fittest individuals. Lines 4 and 5 initializes the algorithm by generating an initial random population of size N. Lines 6–11 iterates over the initial population and evaluates each individual based on the three objective functions F_T, F_S, and F_E (time, offloading survivability, and energy defined in Section IV). Algorithm 2 shows the main idea behind calculating the survivability factor, which gets invoked on line 9. Line 13 ensures elitism in the populations by always selecting the x best individuals and adding them to the new population. Line 14–16 illustrate the process of breeding new offspring by selection and applying crossover and mutation based on their probability factors μ_c and μ_m. In lines 17–26, each individual in the offspring generation is evaluated and checked against the vehicular constraints. First, the objectives are evaluated for each new individual. Next, the individual is checked for constraints violations using Algorithm 3 in line 22. If a solution, contains a violation, the algorithm ignores it and attempts to generate a new solution until a violation-free solution is selected. The entire process is repeated until the termination criteria is met as shown in line 12.

Algorithm 1 Vehicular Resource-Aware Offloading (N, L, μ_c, μ_m)

1: **Input:** Population Size N, Individual Length L, Crossover Rate μ_c, Mutation Rate μ_m
2: **Output:** Set of Fittest Individuals S
3:
4: Set Initial Population Index $k := 0$
5: Generate Random Population P_k of Size N
6: **for** $i \leftarrow 0$ to N **do**
7: Individual $I \leftarrow P_k[i]$
8: Get R::*getRfromFogObserver*(I)
9: Compute Offloading Survivability ***OffloadingSurvivabilityProcessor(I, R)***
10: Evaluate Objective Functions $F_T(I), F_S(I), F_E(I)$
11: **end for**
12: **while** !Termination Criteria **do**
13: Select x Best Solutions from P_k and add them to P_{k+1}
14: Select n Fittest Solutions of P_k using Binary Tournament such that $n = N - x$
15: Crossover μ_c n using SBX Crossover
16: Mutate μ_m n using Polynomial Mutation
17: **for** $i \leftarrow 0$ to N **do**
18: Individual $I \leftarrow P_{k+1}[i]$
19: Get R::*getRfromFogObserver*(I)
20: Compute Offloading Survivability ***OffloadingSurvivabilityProcessor(I, R)***
21: Evaluate Objective Functions $F_T(I), F_S(I), F_E(I)$
22: **while** ***ConstraintsEvaluator(I, R)*** $=$ **false do**
23: Crossover μ_c n using SBX Crossover
24: Mutate μ_m n using Polynomial Mutation
25: **end while**
26: **end for**
27: $K \leftarrow K + 1$
28: **end while**
29: **return** S Fittest Solution From P_k

Algorithm 3 focuses on ensuring that an individual is a valid solution that lacks violations. It expects two parameters; an individual and Requester's ORT. Lines 5–8 assure that the *Requesting Node* will still be in the network range when all Serving Nodes finish executing their chunks. This is done by checking if the remote execution time of an individual I is less than the Requesters ORT R. Lines 9–16 assure that the Serving Node is capable of executing its assigned chunk before it gets out of the Master Nodes range. This is achieved by checking if the node is assigned a number of chunks exceeding the maximum number of allowed chunks that it can execute while it is still within the network range.

On the other hand, the offloading survivability of an individual is calculated in Algorithm 2. The latter expects an individual I and the *Requesting Nodes* ORT R. Briefly, it iterates over all *Serving Nodes* in I and obtains the total execution time of each node n^t. Then, the difference between R and n^t is computed and added into one value for all nodes. Finally, the average value is computed and returned.

Algorithm 2 Offloading Survivability Processor(I, R)

1: **Input:** Individual I and Requester out of range time R
2: **Output:** Offloading Survivability S of Individual I
3:
4: $value \leftarrow -1$
5: **for each** Node $n \in I$ **do**
6: **if** $n \neq Requester$ **then**
7: Get execution time n^t of node n
8: $value \mathrel{+}= (R - n^t)$
9: **end if**
10: **end for**
11: $value \mathrel{/}= I.LENGTH$
12: **return** $value$

Algorithm 3 Constraints Evaluator(I, R)

1: **Input:** Individual I and Requester out of range time R
2: **Output:** true if Individual I contains no violations otherwise false
3:
4: Get remote execution time I^t
5: **if** $I^t \geq R$ **then**
6: $Flag[I] \leftarrow$ **true**
7: **return false**
8: **end if**
9: **for each** Node $n \in I$ **do**
10: Get number of assigned chunks c_n
11: Get the maximum allowed chunks MAX_{c_n}
12: **if** $c_n \geq MAX_{c_n}$ **then**
13: $Flag[I] \leftarrow$ **true**
14: **return false**
15: **end if**
16: **end for**
17: **return true**

TABLE II
TYPE OF DEVICES

Android OS	Android OS
2.5 GHz Quad Core Processor	2.27 GHz Qualcomm Snapdragon 800
2 GB RAM	2 GB RAM
16 GB Storage	32 GB Storage

VI. EXPERIMENTAL RESULTS

In this section, we illustrate our experiments and results. To start with the testbed setup, these experiments are performed on two different type of devices, installed in vehicles, which are described in Table II.

The *ad hoc* vehicular fog we built is composed of ten devices connected through Wi-Fi P2P on Android that uses 802.11 networking standard. These experiments aim to study all of the performance, energy, and number of selected nodes metrics. The response time of the IDS or in other words the time needed to execute the task measure the performance metric. As for the energy, it is the total energy spent on computations and over the network connection. The number of selected nodes refers to the reading-for-serving nodes in the *ad hoc* vehicular fog. We used PowerTutor [22] to monitor the power consumed on computations, device screen, and over

TABLE III
Ad Hoc VEHICULAR FOG SCENARIOS

Composition	Details
7 Critical 3 Idle	The ad-hoc vehicular fog consists of 7 Critical and 3 Idle nodes.
4 Critical 2 Moderate 4 Idle	The ad-hoc vehicular fog consists of 4 Critical, 2 Moderate, and 4 Idle nodes.
10 Idle with 5 Problematic	The ad-hoc vehicular fog consists of 10 Idle nodes. However, 5 of them are problematic nodes having moving high speed compared to the rest.

TABLE IV
VANET EXECUTION SCENARIOS

Scenario	Description
Local	Local execution on Idle Requesting Node.
EQ	Equal Offloading in a vehicular environment.
VIOD	Execution using our Vehicular Intelligent Offloading Distributor.
Best E	Execution using our Vehicular Intelligent Offloading Distributor when Requester Node is in an Idle state and energy is prioritized.
Best T	Execution using our Vehicular Intelligent Offloading Distributor when Requester Node is in an Idle state and time is prioritized.
Best S	Execution using our Vehicular Intelligent Offloading Distributor when Requester Node is in an Idle state and offloading survivability is prioritized.
10% ORT	Each node in the ad-hoc vehicular fog has an ORT value allowing it to serve at most 10% of the data.
30% ORT	Each node in the ad-hoc vehicular fog has an ORT value allowing it to serve at most 30% of the data.
50% ORT	Each node in the ad-hoc vehicular fog has an ORT value allowing it to serve at most 50% of the data.
100% ORT	Each node in the ad-hoc vehicular fog has an ORT value allowing it to serve up to 100% of the data.

the network for computing the energy consumed by a particular task. We also used the power profile [23] on android devices for getting the power consumption of Wi-Fi transmission, power consumed by the CPU when idle, power consumed by active Wi-Fi transmission, and power consumed by the screen. The *Profiler* module is responsible for gathering the CPU, memory usage, and battery level on the device.

Moreover, we provide three different states: *Idle, Moderate*, and *Critical* for each node in order to reflect a diverse real-life environment in our experiments. An *Idle* node is in stand-by not performing any task other than the default running services with a CPU usage between 0 and 10%. A *Moderate* node performs some services, including those of GPS, networking, Bluetooth, etc., with a CPU usage not exceeding 55%. A *Critical* node can attain a very high CPU usage, above 90%, performing several compute-intensive tasks. Accordingly, our *ad hoc* vehicular fog is built using combinations of the aforementioned scenarios. We include three different scenarios of the *ad hoc* vehicular fog depicted in Table III. In the first scenario, we consider an *ad hoc* vehicular fog composed of seven *Critical* and three *Idle* nodes. In the second scenario, we consider the fog to be composed of four *Critical*, two *Moderate*, and four *Idle* nodes. Finally, in the third and last scenario, we take a critical case of the fog by introducing ten *Idle* nodes, five of which are moving in high speeds compared to the rest. In addition, we consider several execution scenarios in which we vary the nodes ability to execute chunks depending on their ORT as illustrated in Table IV. In order to determine a clear conclusion of our experiments, we perform the set of executions on data of sizes 500 kB and 1 MB since the shared files for performing intrusion detection are not big in real-life scenarios. In fact, they contain traces for scanning generated within minutes or hours not more, especially in the vehicular environment with high mobility. On the other hand, we have conducted relevant experiments in previous work in order to study the efficiency of offloading with larger data sets [5]. The results are promising as they positively scaled with the larger data size ranging from 1 to 50 MB. Furthermore, we highlight the importance of one more criterion used within our experiments for representing the mobility effect of the vehicle nodes and survivability of the clustered fog. The *Requester Node's* ORT is of vital importance since it is one of the crucial factors used in the vehicular model. We note that some results from Equal Offloading in the vehicular experiments are not generated in the charts, due to the fact that Equal Offloading will not finish its execution within the requester's ORT. This is not the case in our solution that considers the mobility and eliminates the vehicles going out of range while forming the cluster.

In this work, we are dealing with a Pareto multiobjective optimization problem where the objective functions are conflicting (i.e., execution time or performance, survivability, and energy). Accordingly, there is no single dominating optimal solution and the possible resolutions are somehow not comparable without preferred constraints. In this regard, reducing the weight of any of the objective functions is not possible without increasing at least one of the other ones. This is known as the concept of pareto-optimality or nondominated solutions chosen by the algorithm. In this case, an analysis should be performed on the multiple solutions representing the tradeoffs among the objective functions from which only one will be chosen based on embedded decision-making task enforcing some preferred constraints while considering the profile statuses of the vehicular fog nodes [24]. In this context, we did apply the aforementioned recommended study by providing several experiments exploring the best solutions with tradeoffs while enforcing constraints on one or more of the objective functions. In other words, we have provided a set of scenarios focusing on one or more objective functions through varying their weights (i.e., increase/decrease their values) as illustrated in Figs. 3–8. Moreover, we did compare our proposition to both local execution (i.e., on device) and equal distribution (i.e., load distributed equally to all devices without considering any of the criteria) approaches. We have also conducted in previous work [25], [26] several experiments to

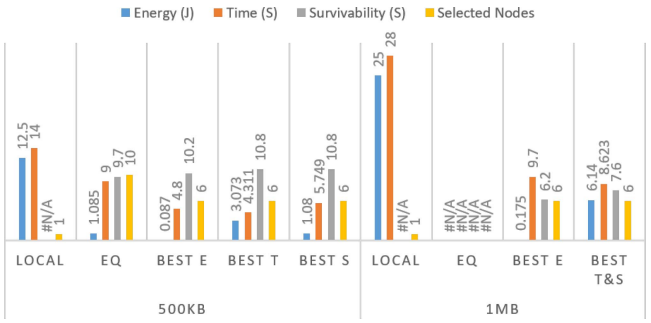

Fig. 3. Evaluation of our proposed solution in an *ad hoc* vehicular fog composed of four critical, two moderate, and three idle nodes.

Fig. 4. Evaluation of our proposed solution in a *ad hoc* vehicular fog composed of seven critical and three idle nodes.

evaluate the overhead of NSGAII, SPEA2, IBEA, MOCell, and SMSEMOA algorithms on mobile devices. In this regard, NSGAII provided better performance and less energy and CPU consumption. In this article, we adapted NSGAII [18] to the context of *ad hoc* vehicular fog to classify solutions based on their Pareto ranking and find a good tradeoff between several conflicting objective functions.

In the below experiments we compare our VIOD module to local execution and equal offloading. We vary the input size between 500 kB and 1 MB for testing under different circumstances. The task execution time and the requester node energy consumption are monitored in each scenario. Moreover, we show the number of nodes used and the offloading survivability value of the fog. We highlight the impact of the survivability factor by the following relation: the higher this value is, the more time is left in the fog under the same state for it to recover from any unexpected activity before the *Requesting Node* leaves the network range. For our VIOD, we show multiple solutions based on prioritizing the different objectives as illustrated in Table IV.

Fig. 3 illustrates the evaluation of our approach in a scenario where the *ad hoc* vehicular fog is composed of four *Critical*, two *Moderate*, and three *Idle* nodes in a vehicular environment. Using 500-kB input data and prioritizing the energy over the two other objectives shows that our module is able to reduce the energy consumption by 99.3% and lower the execution time by 65.7% compared to local execution. Compared to equal offloading, our VIOD module is capable of reducing energy consumption by 91.9%, lowering execution time by 50%, and decreasing the number of used nodes by 40%. On the other hand, prioritizing the time objective leads to 75.4% reduction in energy consumption and a 69.2% decrease in execution time compared to local execution. Compared to Equal Offloading, our module reduces execution time by 55% and lowers the number of selected nodes by 40%, however, it uses 64.6% more energy. Finally, regarding the offloading survivability case, we show that even if it is prioritized over energy and time, our VIOD still outputs good results. Energy is reduced by 91% and execution time by 59% compared to local execution. Compared to Equal Offloading, energy remains the same, the number of used nodes is reduced by 40%, and execution time is decreased by 40.7%. Similar results are shown using 1-MB data when comparing our VIOD to local execution. However, we note that Equal Offloading is not possible in this case because by the time the offloading process finished, the *Requesting Node* had already gone out of the *Master Node's* range. As a result, the offloading decision is not utilized and a final result is not received by the *Requesting Node*. Another important observation is that prioritizing the offloading survivability results in the same solution as that of prioritizing the time objective where energy and execution time are reduced by 75% and 69%, respectively. As a result, we show that our VIOD outperforms Equal Offloading while drastically improving execution time and energy consumption when prioritizing any of the three objectives compared to local execution.

Fig. 4 shows our approach's evaluation using a 500-kB file and an *ad hoc* vehicular fog composed of seven *Critical* and three *Idle* nodes in a vehicular environment. Prioritizing energy over the other two objectives, shows that our module is able to reduce the energy 99.3% and time by 52.6% compared to local execution. Compared to equal offloading, our VIOD module reduced energy consumption by 91.9%, execution time by 51%, and the number of used nodes by 40%. On the other hand, prioritizing the time objective leads to 75.4% reduction in energy consumption and a 65.4% decrease in execution time compared to local execution. Compared to Equal Offloading, our module reduces execution time by 62.8% and number of selected nodes by 40%, however, it consumes 64.6% more energy. Finally, regarding the offloading survivability factor, we show that even if it is prioritized over energy and time, our VIOD still outputs good results. Energy is reduced by 91% and execution time by 59% compared to local execution. Compared to Equal Offloading, energy remains the same, number of used nodes is reduced by 40%, and execution time is decreased by 55.8%. Similar results are shown using 1-MB input data when comparing our VIOD to local execution. However, we note that Equal Offloading is not possible in this case because by the time it finishes, the *Requesting Node* had already gone out of the *Master Node's* range. Another important observation is that prioritizing the offloading survivability results in the same solution as that of prioritizing the time objective. As a result, we show that our VIOD outperforms Equal Offloading while drastically improving execution time and energy consumption when prioritizing any of the three objectives compared to local execution.

Table V engages deeper into illustrating the case of having ten *Idle* nodes five of which are problematic. We suppose a problematic node as a node having high moving speed compared to other nodes in the *ad hoc* vehicular fog. Therefore,

TABLE V
SKIPPING NODES WITH HIGH OUT-OF-RANGE SPEED

Node	0	1	2	3	4	5	6	7	8	9
State	Idle	Idle	Idle	Idle	Idle	Idle	Idle	Idle	Idle	Idle
Problematic	No	Yes	No	Yes	No	Yes	No	Yes	No	Yes
Solution 1	0	0	20	0	20	0	30	0	30	0
Solution 2	20	0	20	0	20	0	20	0	20	0
Solution 3	20	0	10	0	20	0	30	0	20	0

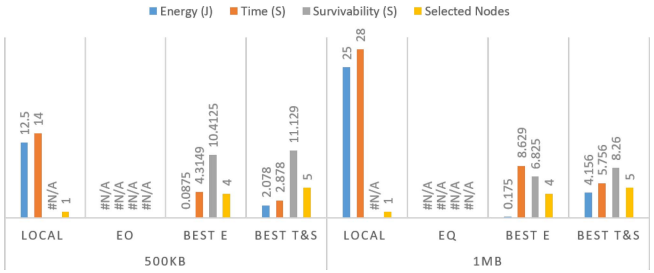

Fig. 5. Evaluation of our proposed solution in an *ad hoc* vehicular fog composed of ten idle nodes where five of which are problematic.

Fig. 6. Evaluation of our proposed solution in an *ad hoc* vehicular fog composed of seven critical and three idle nodes while varying the nodes ORT using a 1-MB file.

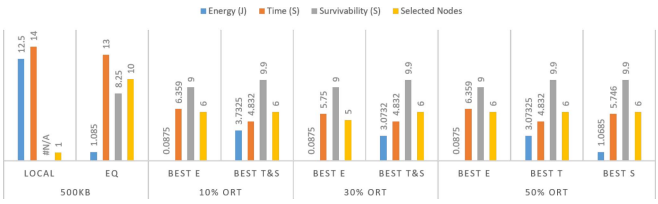

Fig. 7. Evaluation of our proposed solution in an *ad hoc* vehicular fog composed of seven critical and three idle nodes while varying the nodes ORT using a 500-kB file.

it has a relatively low ORT that violates one or both of our vehicular constraints. The first row lists the nodes, the second lists the node's state, and the third shows whether the node is problematic or not. The fourth, fifth, and sixth rows show the solutions obtained using our Intelligent Offloading Distributor adapted to the vehicular environment (VIOD module). Even though the solutions vary depending on the weight given to the objective functions (energy, time, and survivability), all solutions exclude problematic nodes. In other words, the distribution output is different, but none of the problematic nodes are used within the solutions. Therefore, we show that even if the node is in an *Idle* state and can quickly execute offloaded chunks to reduce the execution time, the node must not violate any vehicular constraints for it to be considered in an intelligent solution.

Fig. 5 is devoted to show the results of the case study shown in Table V. In a nutshell, we show the performance of our VIOD module when all nodes are *Idle* and five of them are problematic (i.e., moving fast compared to the rest). As a first observation, we note that Equal Offloading in a vehicular environment is not possible. This is due to the fact that some of the nodes are either unable to execute at least one chunk or unable to return the result on time (i.e., before the requester's ORT passes). Favoring energy over time using a 500-kB file shows that four nodes are used, 99.3% decrease in energy, and 69% decrease in execution time when compared to local execution. Another important observation is that favoring time leads to the same optimal result as favoring survivability. Energy and execution time are reduced by 83% and 79.4%, respectively, while using five nodes of the ten nodes composing the *ad hoc* vehicular fog. Comparable results are achieved using our VIOD module under a 1-MB file. Energy and execution time are reduced by 99.3% and 69%, respectively, while selecting four nodes out of ten when energy is prioritized. Regarding prioritizing time and survivability one optimal solution is obtained where energy is reduced by 83.3% and execution time is decreased by 79.4% while using five out of the ten available nodes. Therefore, we show that our VIOD outputs optimal and very promising results. The results also show that Equal Offloading fails under critical scenarios. These results can be justified by the fact that our approach is taking into account different metrics of time, survivability and energy with priorities relevant to each scenario in order to decide about the best distribution of tasks.

Fig. 6 is dedicated to show the effectiveness of our VIOD module in an *ad hoc* vehicular fog composed of seven *Critical* and three *Idle* nodes while varying the node's ORT allowing it to execute at most 1, 3, and 5 chunks under a 1-MB file. We start by noting that Equal Offloading is impossible under this composition since the result won't be received while the *Requester Node* is still in range. Promising results are achieved in all of the three cases where energy is reduced between 75.4% and 99.3%, execution time is decreased between 54.6% and 65.4% compared to local execution. Moreover, we note that the number of used nodes is reduced by 50% and 60% in most cases. Finally, we show that even when the offloading survivability is prioritized over the other objectives, similar results are obtained. Therefore, we conclude that our VIOD module produces favorable results regardless of the number of chunks that a node can handle. These results can be justified by the fact that the equal distribution would assign the same number of chunks for each node which hence might not be able to finish their execution before going out of the range. However, VIOD tasks into consideration the survivability aspect of the VEC fog when assigning the tasks to each node.

Fig. 7 shows the same set of experiments as Fig. 6 but using 500 kB of data. One major difference between both experiments is that in the latter case, Equal Offloading is a possibility. However, it yields a very high execution time compared to our VIOD module. Equal offloading reduces the execution time by only 7% while our VIOD results show significant improvement up to 65.7%. Another important observation is that the execution time in the Equal Offloading

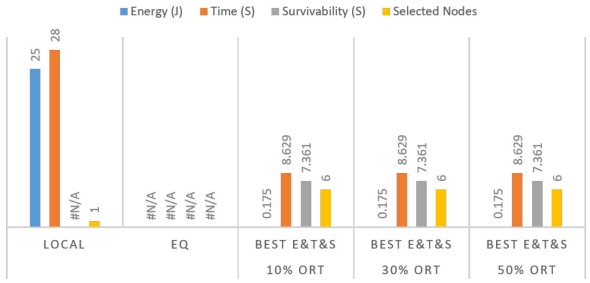

Fig. 8. Evaluation of our proposed solution in an *ad hoc* vehicular fog composed of four critical, three moderate, and four idle nodes while varying the nodes ORT using a 1-MB file.

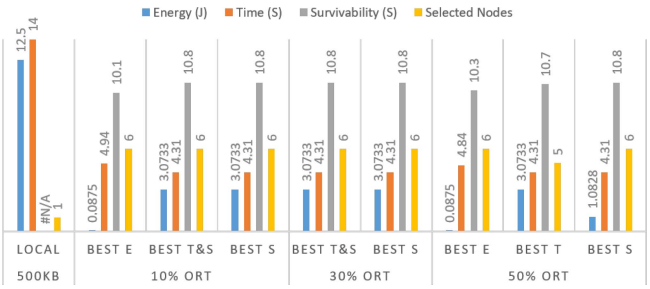

Fig. 9. Evaluation of our proposed solution in an *ad hoc* vehicular fog composed of four critical, three moderate, and four idle nodes while varying the nodes ORT using a 500-kB file.

case is only 1 s less than the *Requesting Node's* ORT. As a result, the fog has only 1 s to resolve any delay or error that could occur in contrast to our VIOD module which gives the fog 7.65 s as a worst case and 9.2 s at best. The figure also shows that our VIOD module is capable of effectively outperforming both Local and Equal Offloading in terms of energy and execution time. For example, the best cases show that energy is reduced by 99.3% and 91.9% while execution time is decreased by 65.7% and 63% compared to Local and Equal Offloading, respectively.

Fig. 8 is dedicated to show the performance of our VIOD model in an *ad hoc* vehicular fog composed of four *Critical*, two *Moderate*, and four *Idle* nodes using a 1-MB file while varying the maximum chunks a node can serve between 1, 3, and 5. We note that Equal Offloading is not feasible under this scenario because the offloading process takes more time than the *Requesting Node's* ORT. Another observation is that in all cases, one solution satisfies all of the three objectives. In other words, our VIOD module is able to generate one optimal solution that minimizes energy and execution time on the device and maximizes the offloading survivability of the fog. Finally, we note that the same optimal solution is obtained in all cases where we varied the node's ORT. Energy and execution time are reduced by 99.3% and 69%, respectively when compared to local execution. The optimal solution uses six out of ten nodes of the *ad hoc* vehicular fog hence saving some resources for other offloading requests. We also note that the fog has an offloading survivability value of 7.3 s indicating that under the same selected nodes, status, and constraints, the fog has 7.3 s to accommodate any delays in the offloading decision and process. Fig. 9 shows comparable results using a 500-kB file. Therefore, we show the effectiveness of our VIOD module using 500 kB and 1-MB file sizes in outputting optimal solutions satisfying the different objective functions according to their weights and priority.

VII. Security Analysis

Like any distributed critical task computations, including security services, offloading IDS services to fog at the edge entails several challenges compared to the centralized cloud framework. However, in the scope of this work, we are concerned with the sustainability and continuity of performing the IDS services while considering the limited resources of the VANET platforms, and availability, communication overhead and latency caused by the cloud framework approaches. In this context, we highlight in the following some related security challenges with potential solutions to address them based on the literature.

1) *Authentication and Trust Management:* According to [27], mutual authentication among peers may be granted by several mechanisms without the need of having a central authentication server. Moreover, Tsai and Lo [28] discussed the problem of authenticating users in a distributed manner for mobile cloud computing by using pairing cryptosystems and secure hardware. The same problem was tackled but for the fog computing paradigm, in which they used a hybrid encryption instead [29]. Other works, such as the ones in [30] focused on authenticating peers according to information extracted from their current location. Such approaches could be adapted to the underlying infrastructures of *ad hoc* vehicular fog paradigms for authenticating the volunteering nodes. Moreover, in order to guarantee reliable communications among the vehicles, master nodes, and corresponding edge devices, the concept of trust might be employed. The idea is to help the master node selecting edge devices that enjoy high trustworthiness to offload intrusion detection tasks in order to ensure that these tasks will be executed in a reliable and well-performing fashion. For this end, objective trust models [31]–[33] can be used. In objective trust, behavior monitoring can be initiated by the master node in order to directly learn the behavior of the different vehicles in its cluster. In subjective trust, the master node can consult other vehicles on the behavior of the edge devices. Thus, vehicles send their recommendations to the master node about the potential trust degree of the edge devices based on their past interactions. Using both objective and subjective trust sources, the master node can then compute an aggregate trust level for each corresponding edge device, which would help it in choosing the appropriate device that will best serve each underlying task.
2) *Network Security:* Trust models can also be backed up with lightweight cryptographic solutions [34], [35], which can be particularly useful to protect the communications among master nodes, edge devices, and fog nodes and also the data pertaining to the intrusion detection tasks at the storage level. Furthermore,

it is possible to exchange session keys by using cryptographic attributes as credentials [27]. Other works addressed the problem of maintaining the interdomain credentials of the trust domains to establish secure channels, such as in [36]. Such solutions can be adapted to the *ad hoc* vehicular edge fog paradigms due to the fact that their requirements are soft and may be altered.

3) *Privacy:* Recently, privacy is one of the excessively addressed topics in the edge computing paradigms. In particular, many data privacy mechanisms have been developed for the mobile cloud computing paradigm, tackling several challenges like the privacy policies enforcement during code and data migration [27], and the peer-to-peer network establishment by concealing the location of adjacent clients [37]. The main requirements of such mechanisms are having the devices interconnected and knowing their physical location. Therefore, it is more likely to invest later on in the privacy mechanisms for collaborative *ad hoc* vehicular fogs. Other mechanisms exist in the literature that tackles directly the vehicular networks and may be adapted to the context of *ad hoc* vehicular fog, in which they make full use of the interconnected local cloudlets concept [38].

VIII. Related Work

In this section, we present a literature review, including several approaches addressing the performance of IDSs on mobile devices, cloud-based IDS, intrusion detection in *ad hoc* networks, and offloading in MEC cloud-enabled vehicular networks.

To start with, cloud-driven intrusion detection techniques are surveyed in [39]. The work also highlights various problems relevant to its implementation. The proposed techniques offer many advantages when it comes to reduction in bandwidth usage, the power needed for processing, and the energy consumer on mobile devices. However, the authors also emphasize on the fact that data should be transferred to the cloud through network communication devices forming potential security breaches where attackers gain access to critical information. Zonouz *et al.* [40] have introduced a framework that provides efficient real-time off device protection through continuous synchronization with an emulated device on the cloud. The framework encompasses a client agent that continuously passes devices input to the cloud where emulated devices are hosted and events from different interfaces of the device are analyzed to ensure periodic backup of the device. More intelligent decisions can be taken in the framework proposed in [41]. It decides whether to execute the detection process locally or offload it for cloud-based analysis along with details regarding the security level and detection algorithm to be applied on the device. The framework includes different intrusion detection engines to enforce security against a wider range of malware. Continuous synchronization between mobile and cloud services is needed.

Although the contributions of previous attempts are significant, yet they are burdened with many limitations. First, they all impose high power consumption for sending traces, events, and inputs over mobile data. Also, mirroring the traffic and data to the cloud overloads the mobile infrastructure and levies extra charges on data usage. Additionally, the usage of remote servers is not free of charge, which imposes additional cost. Furthermore, with existing Wi-Fi technologies, these approaches suffer from intrinsic latency limitation, especially over long-distance communications. These drawbacks have motivated our proposition for *ad hoc* vehicular fog, whereby, to the best of our knowledge, none of the proposed approaches has already considered *ad hoc* vehicular fog for security services.

The only relevant work in this context is the one presented in [42]. It introduces an elastic computing platform for smart device. The proposed platform relies not only on infrastructure-based cloud but also encompasses an *ad hoc* virtual cloud to achieve higher scalability. Smart devices in the vicinity connected via wireless radio, such as Bluetooth form the *ad hoc* virtual cloud and cooperate to accomplish particular offloaded tasks. Whereas intensive computations are handled by the infrastructure-based cloud. Nonetheless, the authors in this work did not emphasize on the *ad hoc* part or provided technical details with respect to the offloading and cooperation aspect between the nodes. Additionally, no experiments have been discussed in terms of performance and cooperation execution on clustered devices.

In another context, we present a state-of-the-art review of different approach that offers a sort of protection in *ad hoc* networks. Different approaches are taken for intrusion detection in mobile *ad hoc* networks (MANETs). While some techniques involve an IDS agent on every single node, others implement the IDS on existing cluster heads which reduces the energy consumption. While an additional overhead is observed on cluster heads, appropriate load distribution methods can be applied to extend the node's lifetime. Increased detection rate and significant reduction in false positive was shown by the proposed work in [43]. The latter proposes a dynamic intrusion detection solution in MANETs, which by leveraging genetic algorithms and artificial immune system, is able to adapt to transformations in the network topology. In [44], a dynamic IDS in MANETs has been presented aiming to increase the detection rate and reduce the false positives. Through genetic algorithm and artificial immune system, this approach is capable of adapting changes in the network topology. Ramkumar *et al.* [45] have presented a mechanism for intrusion detection in MANETs relying on a master cluster head. Each cluster head monitors the nodes behavior within its cluster aiming to handle intracluster detection and elimination of malicious nodes. However, these approaches differ from our work in the sense that they tackle IDSs in routing protocols as for the detection of misbehaving nodes. These works are capable of detecting malicious nodes, while on the other hand, our work focuses on IDS monitoring system activities within a node.

Finally, in the field of offloading in MEC cloud-enabled vehicular networks, few recent approaches have been proposed to offload network and computation tasks to edge servers and nearby devices. Zhang *et al.* [12] advanced a cloud-based

MEC offloading approach for vehicular networks in order to offload efficiently the tasks to the MEC servers by either directly uploading them, or through predictive relay transmissions. Xiao *et al.* [13] and Sun *et al.* [46] proposed a learning task offloading algorithm using the multiarmed bandit theory for the sake of optimizing the delay performance of tasks. Gu *et al.* [47] attempted to reduce the overall energy consumption while taking into consideration the delay requirements by formulating the task assignment problem as one-to-many matching game. Al-Badarneh *et al.* [48] proposed network-based stations for embedding MEC services. Atallah *et al.* [49] derived and presented a polynomial-time solutions through heuristic strategies for efficient scheduling of electric vehicles at charging stations. However, to the best of our knowledge, none of the current approaches has yet considered the offloading of security services in a MEC fog-based vehicular environment and provided solutions relying on infrastructure-less architectures.

IX. Conclusion

In this article, we proposed a VEC fog-enabled scheme that intelligently and efficiently offloads intrusion detection tasks to federated vehicles to be executed with minimal latency, while accounting for the high mobility of vehicles, availability of resources, and survivability of the clustered vehicles. Our approach is particularly suitable for security assurance in next-generation AI-powered ITS that require efficient and lightweight security solutions. The proposed approach has been experimentally evaluated with respect to three other approaches (i.e., local offloading, equal offloading, and intelligent offloading) and on resource-constrained devices reflecting actual *ad hoc* vehicular fog environment. The experimental results show that our approach minimizes the energy consumption and execution time of the intrusion detection tasks, while increasing the offloading survivability time under different types of nodes (i.e., idle, moderate, and critical), different ORT percentages on the nodes (i.e., 10%, 30%, 50%, and 100%) and different intrusion detection data file sizes (i.e., 500 kB and 1 MB). On the other hand, few aspects related to the selection of participants in the federation of vehicles are still missing in the proposed framework. As future work, stable vehicular fog federation formation, vehicular federated learning, trust-based vehicular federation, trust and security relation among vehicles, and efficient aggregation for federated intrusion detection may constitute promising research directions for extending our scheme.

References

[1] M. Mohammadi, A. Al-Fuqaha, S. Sorour, and M. Guizani, "Deep learning for IoT big data and streaming analytics: A survey," *IEEE Commun. Surveys Tuts.*, vol. 20, no. 4, pp. 2923–2960, 4th Quart., 2018.

[2] P. Fabian, A. Rachedi, and C. Guéguen, "Programmable objective function for data transportation in the Internet of Vehicles," *Trans. Emerg. Telecommun. Technol.*, vol. 31, no. 5, 2020, Art. no. e3882.

[3] W. Li and H. Song, "ART: An attack-resistant trust management scheme for securing vehicular ad hoc networks," *IEEE Trans. Intell. Transp. Syst.*, vol. 17, no. 4, pp. 960–969, Apr. 2016.

[4] M. S. Al-Kahtani, "Survey on security attacks in vehicular ad hoc networks (VANETs)," in *Proc. 6th Int. Conf. Signal Process. Commun. Syst.*, Gold Coast, QLD, Australia, 2012, pp. 1–9.

[5] T. Dbouk, A. Mourad, H. Otrok, H. Tout, and C. Talhi, "A novel ad-hoc mobile edge cloud offering security services through intelligent resource-aware offloading," *IEEE Trans. Netw. Service Manag.*, vol. 16, no. 4, pp. 1665–1680, Dec. 2019.

[6] S. Jošilo and G. Dán, "Selfish decentralized computation offloading for mobile cloud computing in dense wireless networks," *IEEE Trans. Mobile Comput.*, vol. 18, no. 1, pp. 207–220, Jan. 2019.

[7] A. Gharaibeh, A. Khreishah, M. Mohammadi, A. Al-Fuqaha, I. Khalil, and A. Rayes, "Online auction of cloud resources in support of the Internet of Things," *IEEE Internet Things J.*, vol. 4, no. 5, pp. 1583–1596, Oct. 2017.

[8] T. Mekki, I. Jabri, A. Rachedi, and M. ben Jemaa, "Vehicular cloud networks: Challenges, architectures, and future directions," *Veh. Commun.*, vol. 9, pp. 268–280, Jul. 2017.

[9] T. Mekki, I. Jabri, A. Rachedi, and M. B. Jemaa, "Vehicular cloud networking: Evolutionary game with reinforcement learning-based access approach," *Int. J. Bio-Inspired Comput.*, vol. 13, no. 1, pp. 45–58, 2019.

[10] M. Khabbaz, "Modelling and analysis of a novel vehicular mobility management scheme to enhance connectivity in vehicular environments," *IEEE Access*, vol. 7, pp. 120282–120296, 2019.

[11] H. Liu, F. Eldarrat, H. Alqahtani, A. Reznik, X. de Foy, and Y. Zhang, "Mobile edge cloud system: Architectures, challenges, and approaches," *IEEE Syst. J.*, vol. 12, no. 3, pp. 2495–2508, Sep. 2018.

[12] K. Zhang, Y. Mao, S. Leng, Y. He, and Y. Zhang, "Mobile-edge computing for vehicular networks: A promising network paradigm with predictive off-loading," *IEEE Veh. Technol. Mag.*, vol. 12, no. 2, pp. 36–44, Jun. 2017.

[13] L. Xiao, W. Zhuang, S. Zhou, and C. Chen, "Learning while offloading: Task offloading in vehicular edge computing network," in *Learning-based VANET Communication and Security Techniques*. Cham, Switzerland: Springer, 2019, pp. 49–77.

[14] I. Jabri, T. Mekki, A. Rachedi, and M. B. Jemaa, "Vehicular fog gateways selection on the Internet of Vehicles: A fuzzy logic with ant colony optimization based approach," *Ad Hoc Netw.*, vol. 91, Aug. 2019, Art. no. 101879.

[15] H. Sami and A. Mourad, "Dynamic on-demand fog formation offering on-the-fly IoT service deployment," *IEEE Trans. Netw. Service Manag.*, vol. 17, no. 2, pp. 1026–1039, Jun. 2020.

[16] H. Sami, A. Mourad, and W. El-Haj, "Vehicular-obus-as-on-demand-fogs: Resource and context aware deployment of containerized micro-services," *IEEE/ACM Trans. Netw.*, vol. 28, no. 2, pp. 778–790, Apr. 2020.

[17] A. Rachedi and H. Badis, "BadZak: An hybrid architecture based on virtual backbone and software defined network for Internet of Vehicles," in *Proc. IEEE Int. Conf. Commun. (ICC)*, Kansas City, MO, USA, 2018, pp. 1–7.

[18] K. Deb, A. Pratap, S. Agarwal, and T. Meyarivan, "A fast and elitist multiobjective genetic algorithm: NSGA-II," *IEEE Trans. Evol. Comput.*, vol. 6, no. 2, pp. 182–197, Apr. 2002.

[19] O. A. Wahab, H. Otrok, and A. Mourad, "Vanet QoS-OLSR: QoS-based clustering protocol for vehicular ad hoc networks," *Comput. Commun.*, vol. 36, no. 13, pp. 1422–1435, 2013.

[20] M. Norton and D. Roelker, *Snort 2.0: Hi-Performance Multi-Rule Inspection Engine*, Sourcefire Netw. Security Inc, Columbia, MD, USA, 2002.

[21] T. Lust and J. Teghem, "The multiobjective multidimensional knapsack problem: A survey and a new approach," *Int. Trans. Oper. Res.*, vol. 19, no. 4, pp. 495–520, 2012.

[22] L. Zhang *et al.*, "Accurate online power estimation and automatic battery behavior based power model generation for smartphones," in *Proc. 8th IEEE/ACM/IFIP Int. Conf. Hardw./Softw. Codesign Syst. Synth.*, Scottsdale, AZ, USA, 2010, pp. 105–114.

[23] *Power Profile for Android*, Android, Mountain View, CA, USA, 2020. [Online]. Available: https://source.android.com/devices/tech/power/

[24] R. Jena, "Multi objective task scheduling in cloud environment using nested PSO framework," *Procedia Comput. Sci.*, vol. 57, pp. 1219–1227, Jan. 2015.

[25] H. Tout, C. Talhi, N. Kara, and A. Mourad, "Selective mobile cloud offloading to augment multi-persona performance and viability," *IEEE Trans. Cloud Comput.*, vol. 7, no. 2, pp. 314–328, Apr.–Jun. 2019.

[26] H. Tout, C. Talhi, N. Kara, and A. Mourad, "Smart mobile computation offloading: Centralized selective and multi-objective approach," *Expert Syst. Appl.*, vol. 80, pp. 1–13, Sep. 2017.

[27] R. Roman, J. Lopez, and M. Mambo, "Mobile edge computing, fog et al.: A survey and analysis of security threats and challenges," *Future Gener. Comput. Syst.*, vol. 78, pp. 680–698, Jan. 2018.

[28] J.-L. Tsai and N.-W. Lo, "A privacy-aware authentication scheme for distributed mobile cloud computing services," *IEEE Syst. J.*, vol. 9, no. 3, pp. 805–815, Sep. 2015.

[29] I. Stojmenovic, S. Wen, X. Huang, and H. Luan, "An overview of fog computing and its security issues," *Concurrency Comput. Pract. Exp.*, vol. 28, no. 10, pp. 2991–3005, 2016.

[30] S. Xu, E. P. Ratazzi, and W. Du, "Security architecture for federated mobile cloud computing," in *Proc. Mobile Cloud Security Conf.*, 2016.

[31] H. Otrok, A. Mourad, J.-M. Robert, N. Moati, and H. Sanadiki, "A cluster-based model for QoS-OLSR protocol," in *Proc. IEEE 7th Int. Wireless Commun. Mobile Comput. Conf.*, Istanbul, Turkey, 2011, pp. 1099–1104.

[32] N. Moati, H. Otrok, A. Mourad, and J.-M. Robert, "Reputation-based cooperative detection model of selfish nodes in cluster-based QoS-OLSR protocol," *Wireless Pers. Commun.*, vol. 75, no. 3, pp. 1747–1768, 2014.

[33] O. A. Wahab, J. Bentahar, H. Otrok, and A. Mourad, "Optimal load distribution for the detection of VM-based DDOS attacks in the cloud," *IEEE Trans. Services Comput.*, vol. 13, no. 1, pp. 114–129, Jan./Feb. 2020.

[34] J. Ni, K. Zhang, X. Lin, and X. S. Shen, "Securing fog computing for Internet of Things applications: Challenges and solutions," *IEEE Commun. Surveys Tuts.*, vol. 20, no. 1, pp. 601–628, 1st Quart., 2018.

[35] R. Yang, Q. Xu, M. H. Au, Z. Yu, H. Wang, and L. Zhou, "Position based cryptography with location privacy: A step for fog computing," *Future Gener. Comput. Syst.*, vol. 78, pp. 799–806, Jan. 2018.

[36] H. M. Pimentel, S. Kopp, M. A. Simplicio, Jr, R. M. Silveira, and G. Bressan, "OCP: A protocol for secure communication in federated content networks," *Comput. Commun.*, vol. 68, pp. 47–60, Sep. 2015.

[37] H. Zhang, N. Yu, and Y. Wen, "Mobile cloud computing based privacy protection in location-based information survey applications," *Security Commun. Netw.*, vol. 8, no. 6, pp. 1006–1025, 2015.

[38] X. Huang, R. Yu, J. Kang, N. Wang, S. Maharjan, and Y. Zhang, "Software defined networking with pseudonym systems for secure vehicular clouds," *IEEE Access*, vol. 4, pp. 3522–3534, 2016.

[39] Z. Inayat, A. Gani, N. B. Anuar, S. Anwar, and M. K. Khan, "Cloud-based intrusion detection and response system: Open research issues, and solutions," *Arabian J. Sci. Eng.*, vol. 42, no. 2, pp. 399–423, 2017.

[40] S. Zonouz, A. Houmansadr, R. Berthier, N. Borisov, and W. H. Sanders, "Secloud: A cloud-based comprehensive and lightweight security solution for smartphones," *Comput. Security*, vol. 37, pp. 215–227, Sep. 2013.

[41] D. Damopoulos, G. Kambourakis, and G. Portokalidis, "The best of both worlds: A framework for the synergistic operation of host and cloud anomaly-based ids for smartphones," in *Proc. 7th Eur. Workshop Syst. Security (EuroSec'14)*, New Yark, NY, USA, Apr. 2014, pp. 1–6.

[42] W. Zhang, Y. Wen, J. Wu, and H. Li, "Toward a unified elastic computing platform for smartphones with cloud support," *IEEE Network*, vol. 27, no. 5, pp. 34–40, Sep./Oct. 2013.

[43] F. Barani, "A hybrid approach for dynamic intrusion detection in ad hoc networks using genetic algorithm and artificial immune system," in *Proc. IEEE Iran. Conf. Intell. Syst. (ICIS)*, Bam, Iran, 2014, pp. 1–6.

[44] A. A. Korba, M. Nafaa, and Y. Ghamri-Doudane, "Anomaly-based intrusion detection system for ad hoc networks," in *Proc. IEEE 7th Int. Conf. Netw. Future (NOF)*, Buzios, Brazil, 2016, pp. 1–3.

[45] P. Ramkumar, V. Vimala, and G. S. Sundari, "Homogeneous and heterogeneous intrusion detection system in mobile ad hoc networks," in *Proc. IEEE Int. Conf. Comput. Technol. Intell. Data Eng. (ICCTIDE'16)*, Kovilpatti, India, 2016, pp. 1–5.

[46] Y. Sun, X. Guo, S. Zhou, Z. Jiang, X. Liu, and Z. Niu, "Learning-based task offloading for vehicular cloud computing systems," in *Proc. IEEE Int. Conf. Commun. (ICC)*, Kansas City, MO, USA, 2018, pp. 1–7.

[47] B. Gu, Y. Chen, H. Liao, Z. Zhou, and D. Zhang, "A distributed and context-aware task assignment mechanism for collaborative mobile edge computing," *Sensors*, vol. 18, no. 8, p. 2423, 2018.

[48] J. Al-Badarneh, Y. Jararweh, M. Al-Ayyoub, R. Fontes, M. Al-Smadi, and C. Rothenberg, "Cooperative mobile edge computing system for vanet-based software-defined content delivery," *Comput. Elect. Eng.*, vol. 71, pp. 388–397, Oct. 2018.

[49] R. F. Atallah, C. M. Assi, W. Fawaz, M. H. K. Tushar, and M. J. Khabbaz, "Optimal supercharge scheduling of electric vehicles: Centralized versus decentralized methods," *IEEE Trans. Veh. Technol.*, vol. 67, no. 9, pp. 7896–7909, Sep. 2018.

Azzam Mourad (Senior Member, IEEE) received the Ph.D. degree in ECE from Concordia University, Montreal, QC, Canada, in 2009.

He is currently an Associate Professor of computer science with Lebanese American University, Beirut, Lebanon, and also an Affiliate Associate Professor with the Software Engineering and IT Department, École de Technologie Supérieure, Montreal.

Dr. Mourad has served/serves as an Associate Editor for the IEEE TRANSACTION ON NETWORK AND SERVICE MANAGEMENT, IEEE NETWORK, the IEEE OPEN JOURNAL OF THE COMMUNICATIONS SOCIETY, *IET Quantum Communication*, and IEEE COMMUNICATIONS LETTER. He has also served/serves as the General Chair of IWCMC2020, the General Co-Chair of WiMob2016, and the Track Chair, a TPC member, and a reviewer of several prestigious journals and conferences.

Hanine Tout received the M.Sc. degree in computer science from Lebanese American University, Beirut, Lebanon, in 2014, and the Ph.D. degree in software engineering from the École de Technologie Supérieure (ETS), Montreal, QC, Canada, in 2018.

She is a Postdoctoral Fellow with ETS and Ericsson Canada, Mississauga, ON, Canada, where she is leading two industrial projects in the areas of AI, federated learning, machine learning, security, 5G, and cloud-native IMS.

Dr. Tout is a TPC member and reviewer of prestigious conferences and journals.

Omar Abdel Wahab received the M.Sc. degree in 2014, and the Ph.D. degree in information and systems engineering from Concordia University, Montreal, QC, Canada, in 2018.

He is an Assistant Professor with the Department of Computer Science and Engineering, University of Quebec at Outaouais, Gatineau, QC, Canada. His current research activities are in the areas of artificial intelligence, cybersecurity, cloud computing, and big data analytics.

Dr. Wahab is a TPC member of several prestigious conferences and reviewer of several highly ranked journals.

Hadi Otrok (Senior Member, IEEE) received the Ph.D. degree in ECE from Concordia University, Montreal, QC, Canada, in 2009.

He holds an Associate Professor position with the Department of ECE, Khalifa University of Science and Technology, Abu Dhabi, UAE, an Affiliate Associate Professor with the Concordia Institute for Information Systems Engineering, Concordia University, and an Affiliate Associate Professor with the Electrical Department, École de Technologie Supérieure (ETS), Montreal. His research interests include the domain of computer and network security, crowd sensing and sourcing, *ad hoc* networks, and blockchain.

Dr. Otrok is an Associate Editor of *Ad Hoc Networks* (Elsevier) and IEEE NETWORKS. He served in the editorial board of IEEE COMMUNICATION LETTERS. He co-chaired several committees at various IEEE conferences.

Toufic Dbouk received the M.Sc. degree in computer science from Lebanese American University, Beirut, Lebanon, in 2017.

He is currently a Senior Software Engineer with Samsung Electronics America, Ridgefield Park, NJ, USA. His research interest is in mobile computing, computation offloading, and intrusion detection systems.