Detecting Faulty and Malicious Vehicles Using Rule-based Communications Data Mining

Jihene Rezgui
Department of Electrical and Computer Engineering
Université de Sherbrooke
Sherbrooke, Canada
Jihene.Rezgui@usherbrooke.ca

Soumaya Cherkaoui
Department of Electrical and Computer Engineering
Université de Sherbrooke
Sherbrooke, Canada
Soumaya.Cherkaoui@usherbrooke.ca

***Abstract*—The reliability of most safety applications that are based on vehicular communications, depends in turn on the reliability of data received by each vehicle from its neighbors. Routine messages exchanged in Vehicular Ad hoc Networks (VANETs) include crucial information for safety applications such as direction, position, etc. A vehicle failure and/or a malicious vehicle transmitting false information may affect the data collection scheme and cause a disturbance for safety applications. In such a scenario, (1) the faulty/malicious vehicle should be detected rapidly and (2) routine messages exchange should be updated in consequence. To be able to detect the faulty/malicious vehicle, we developed a mechanism that collects, at a single vehicle, data regarding each neighbour transmission, and extracts the temporal correlation rules between vehicles implicated in transmissions in the neighbourhood. With the mechanism, called VANETs Association Rules Mining (VARM), a mining process will take place during a-priori constant historical period. The associations rules formulated during the mining process will be used to detect a faulty or malicious vehicle, i.e., a vehicle which is not correlated with vehicles in the neighbourhood following these rules. To react after this kind of anomaly detection, an 1:N technique is used as a protection for re-establishing the accuracy of the data collection process between vehicles communicating in the neighbourhood. Simulation results demonstrate the efficiency of the VARM scheme.**

Index Terms*— *VANETs, safety messages, faulty/malicious vehicles, association rules.

I. INTRODUCTION

Vehicular Ad hoc Networks (VANETs) are considered as a key technology for bringing more safety on the road. In VANETs safety applications, each vehicle periodically receives information from neighboring vehicles, in its transmission range, about their respective current state including position, direction, etc. This is performed by way of *routine* heartbeat safety messages sent every 10ms as per DSRC standard [21]. In most cases, the reliability of these safety applications relies on information gathered from sensors in one or more surrounding vehicles; information that can potentially be faulty or inaccurate. The problem can further be worsened with the possible presence of malicious vehicles intentionally injecting false information which can impact the vehicle perception of its environment.

In order to ensure the timeliness of safety related data as a way to ensure its pertinence to safety applications, a lot of work has been done to reduce the delay and increase the packet receptions. Proposed schemes include MAC layer solutions for backoff algorithm improvement [12-14] and communication rate and/or power adjustment strategies. However, these works are only concerned with the timeliness of information and are unable to detect and correct the presence of false or faulty data. Also, these studies do not consider the fact that a vehicle, which we will call throughout the paper "*malicious"* vehicle, could provide wrong information. This is in contrast with a "*faulty"* vehicle which is a vehicle transmitting faulty information.

In this work we were interested in building and mining association rules based on routine messages received by a vehicle. These messages typically help capture spatial relations, i.e. direction, position, etc. but also temporal relations between vehicles. We propose a novel scheme called VANET Association Rules Mining (VARM) that collects, at a single vehicle, data regarding each neighbor transmission, and extracts the temporal correlation rules between vehicles implicated in transmissions in the neighborhood in order to detect faulty or malicious vehicles.

VARM generates association rules correlating vehicles from vehicle communication events. The associations rules will help, first, analyze the behavior of surrounding vehicles and, second, detect faulty or malicious ones. Building a rule-based communications data mining scheme for vehicles presents several challenges. A formal modeling of vehicles behavior around a particular vehicle is the first challenge. Collecting data (routine messages) from vehicles and proposing a distributed data extraction mechanism is a second challenge. Lastly, given the frequency and dynamic aspect of vehicles, a third challenge is to adapt/design an efficient structure that compresses the collected data from vehicles and allows mining itemsets to derive association rules. Itemsets are defined as a group of items each of which represents a transaction. A transaction is in turn defined as an event received from a vehicle. An example of such a rule is ($m_1 m_2 \Rightarrow m_4, 98\ percent, \alpha$), which means that there is a 98% chance that an event will occur in node m_4 within α units of time when an event is received from $m_1 m_2$ nodes. The association rules allow predicting sources of future events. In the previous example, if no event occurs in m_4, and given the corresponding percentage we can conclude that m_4 node is a faulty vehicle.

The reminding of the paper is organized as follows: Section II briefly discusses related work in the field. An overview of VARM is described in Section III. VARM architecture is

detailed in Section IV. Section V presents a performance study and finally, Section VI concludes the paper.

II. Related Work

To the best of our knowledge, this work is the first to propose communications data mining to detect vehicles anomalies or malicious behavior. For detecting faulty vehicles in VANETs, only few studies [9, 11] have been done. In [11], the authors proposed a model-free approach for detecting anomalies in un-manned autonomous vehicles. Their approach is based on the sensor readings.

However, many researchers [16]–[20] have recently studied security issues to detect malicious VANETs. In [16], the authors proposed a "light-weight" and scalable framework to detect malicious behavior. Studies in [17]–[20] proposed to preload each vehicle, during vehicle registration, with a pool of pseudonyms generated by some government entity. The pseudonyms are used to hide a vehicle's unique identifier. When a vehicle needs to report an event, it randomly picks one pseudonym and signs the message with it, using public key cryptography. This makes it easy to verify the identity of a malicious vehicle should there be need to.

In the general area of wireless networking, however, several works have been done for detecting faulty or malicious nodes using data mining. This is specially the case for wireless sensor networks (WSNs). The authors in [6] investigated the problem of mining associations that exist between sensor values in a stream of data reported from a WSN. They proposed a data model in which sensor nodes are assumed to take values from a finite discrete number of values. The time is divided into equal-sized intervals, and a report from the sensor reading is taken whenever there is a sensor reading update. Each extract context is associated with a weight value that indicates how many intervals this reading is valid. The review of the main techniques that have been introduced for generating association rules in WSNs can be found in [7]. Furthermore, in [7], the authors addressed the problem of extracting data from WSNs for mining patterns regarding the sensor nodes themselves. The data used in the mining process is metadata, describing the nodes activities. They proposed a compressed structure called Positional Lexicographic Tree (PLT) and its mining algorithm has been compared to the FP-Growth [5] algorithm, a well-known data mining algorithm for generating association rules.

Classical approaches for the extraction of implicit knowledge suffer from the huge number of potentially interesting correlations that can be drawn from a dataset (a set of data). In order to limit the number of the reported rules, while conserving the "informativeness" property, the mathematical foundations of the Formal Concept Analysis (FCA), provided a battery of results, known as generic bases of association rules [3]. It is noteworthy to mention that most recent proposed approaches advocate the use of advanced data structures, essentially based on trees, to either store compactly input datasets or to store partial outputs (e.g.,[6, 7]).

In this paper, we propose 1) to use mining to detect faulty vehicles or malicious vehicles sending false information and 2) to improve the storage and mining to react to positive detections. To the best of our knowledge, no studies have addressed the problem of extracting data in VANETs and mining itemsets to derive association rules regarding vehicles activities.

III. Overview of VARM

As a consequence of faults (natural or malicious), VANETs-based general services might perform inefficiently. Faulty information that is used beyond the vehicle that is generating it may have even more severe impacts on safety applications using VANETs. In this paper, we propose an extraction mechanism called VARM for detecting anomalies by meaning of maintaining information about relationship between vehicles. For that, with VARM, each vehicle *i* stores the data received from routine messages in a temporary transaction database T_i (see Table II) that will be translated later to a compact representation (see Fig.3). VARM is able to derive generic bases of association rules regarding an event that can be produced by a vehicle. To achieve this purpose, VARM uses a tree-based data structure called Itemset-tree. The Itemset-tree extends the idea proposed by the authors of FP-Tree [4] and Cats [5] of using specific structures to improve storage compression and allow frequent itemset mining without "explicit" candidate itemset generation steps. Next, VARM adapts the proposed Divide and Conquer algorithm [1, 2] to extract frequent closed itemsets with their associated minimal generators (key itemsets, see Table I). Consequently, VARM is able to detect anomalies after analyzing such rules.

The overall process performed by VARM can be summarized as follows: (i) A formulation of vehicle transactional events (ii) The construction of the Itemset-tree for vehicles events, (iii) The construction of the local ordered structures and generation of local associations rules, (iv) The merging of the local generic association rules to derive global ones, (v) The analysis of the associations rules to detect faulty or/and malicious vehicles, (vi) The use of a predesigned protection 1:N, to update relationships between vehicles.

In the following, the terms transaction database and extraction context will be used interchangeably. Throughout this paper, we will consider the notations shown in Table I.

TABLE I. List of Symbols/Parametres

Itemset (I) $I = \{M_1, M_2, \ldots M_k\} \subset M$	*I* represents a set of vehicles.
Extraction context *T*	Transaction Database.
Support (I)	The number of lines in the transaction database *T* containing *I* or the relative appearance frequency of *I*.
Frequent itemset	The support (*I*) is greater than or equal to the given minimum support (*min_supp*).
Closed itemset	The maximal set of common items with the set of objects (see example a) in section IV.D.2).
N_{TS}	The episode's time slot: slot number.
$M = \{M_1, M_2, \ldots M_m\}$	A set of vehicles.
Episode	A couple $e(N_{TS}, I)$.

Association rule r	Relationship between itemsets as in: *r*: $X \Rightarrow (Y - X)$, in which *X* and *Y* are frequent itemsets, and $X \subset Y$.
X	Antecedent of rule *r*.
(Y-X)	Conclusion of rule *r*.
Confidence	$conf(r) = \frac{\sup port(Y)}{\sup port(X)}$
Conf(r) = 1	*r* is called exact association rule (ER).
Conf(r) < minconf (minimal threshold of confidence)	*r* is an approximate association rule (AR).
α	The size of each time slot where $\alpha = t_i - t_{i-1}$
$\chi = t_n - t_1$	χ is the historical period defined during the extraction process.
$\{t_1, t_2, \ldots, t_n\}$	The time is divided into equal-sized slots.
Generator (g)	Key itemset, $\gamma(g) = I$ *and* $\neg\exists\ g_1 \subset g$ *that* $\gamma(g_1) = I$
γ	Closure operator : Let consider $(E, \leq)$ a partial ordered set. Application γ of $(E, \leq)$ in $(E, \leq)$ is a closed operator, if it respects the three following properties for every sub-set $S, S' \subset E$ 1. Isotony: $S \leq S' \Rightarrow \gamma(S) \leq \gamma(S')$ 2. Extensivity : $S \leq \gamma(S)$ 3. Idempotence : $\gamma(\gamma(S)) = \gamma(S)$

IV. ARCHITECTURE OF VARM SCHEME

A. Problem Formulation

We are interested in vehicles activities. Let $M = \{M_1, M_2, \ldots M_m\}$ be a set of vehicles and $I = \{M_1, M_2, \ldots M_k\} \subset M$ an *itemset* of vehicles. We assume that the time is divided into equal-sized slots $\{t_1, t_2, \ldots, t_n\}$ such that $\chi = t_n - t_1$ where χ is a chosen historical period and α is the size of each time slot where $\alpha = t_i - t_{i-1}$.

We view an occurrence *e* of an event as a triplet $e = (N_{TS}, M, Ev)$, where N_{TS} is the episode's time slot, *M* is the vehicle sender of the event, and *Ev* is the event the event included in the routine message such as direction. However, in our context when each vehicle extracts its own transaction database, it carries on neighboring vehicles with same value of the event and which are occurred in the same time slots. Therefore the episode is restricted to a couple $e(N_{TS}, I)$, where *I* is an itemset (a set of vehicles which sent messages in the same time slot and have the same event value).

Fig. 1 shows the conceptual operation of VARM. It is comprised of the following six major components: (1) the data extraction module, (2) the representation structure module, (3) the closed frequent itemsets mining module, (4) the association rules computing module, (5) the detection module (analysis), (6) the decision module (Predesigned protection).

(1) The extraction data module: according to pre-defined features set (historical period, time slot duration), this module collects network activities received by a given vehicle.

(2) The representation structure module: this module represents the extracted data (transaction databases) in compact structure called Itemset-tree.

(3) The closed frequent itemsets mining module*:* this module has as input an Itemset-tree; it mines closed itemsets and their associated minimal generators to generate ordered structures.

(4) The association rules computing module: this module applies definitions to extract minimal exact and approximate association rules within their associated confidence and support levels.

(5) The detection module: this module interprets the temporal correlation activities between vehicles using the association rules and their respective confidences computed in the previous module, thus allowing faulty vehicles detection.

(6) The decision module: when the support and confidence level of the rule containing the suspected faulty vehicle are higher than the pre-defined thresholds, VARM reacts immediately by re-considering older relationships through the technique *1:N*.

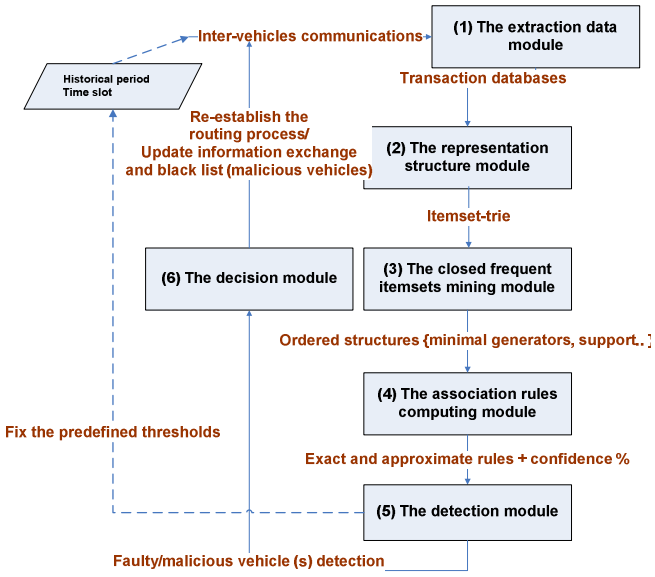

Fig. 1. The Proposed Architecture Model (VARM)

B. Distributed Extraction Mechanism

VARM is interested in capturing the temporal relationships between vehicles based on their activities in common intervals. For that, it starts the process of a data extraction that is implemented in each vehicle as follows:

- Every M_i ($i \in [1..n]$) has as mining parameters: the historical period χ and the time slot α.

- M_i broadcasts these parameters to vehicles in its range using routine messages.
- Every vehicle *in the range* keeps track of the time and checks if any event happened before the end of the time slot.
- If yes, it sends its identifier and the time slot number N_{TS}. Therefore, M_i will create an episode with and additional delay $t+k$ and will store it in the transaction database T_i as shown in Fig.2.

The example of result of this data extraction stage in each M_i is illustrated in Table II.

C. Running Example

Let us consider $M = \{M_1, M_2, \ldots M_6\}$ be a set of vehicles within the range of each other. The historical period χ and the time slot α are set respectively to 50 and 10 second. We assume that the extraction process in every M_i starts at 08:00. In the end of the fourth episode, vehicles $m_2m_3m_6$ send the messages, respectively, (4, m_2), (4, m_3) and (4, m_6) to their *range*. At time (08:40+k), M_2 formulates the fourth episode as e (4, $m_2m_3m_6$) and stores it in the transaction database as shown in Fig.2 and Table II. Periodically the process is repeated at the end of each time slot until the end of the historical period. This example will be used throughout the paper.

D. Itemset-Tree Data Structure

1) Motivation

To detect faulty and/or malicious vehicles, diverse schemes could be applied in VANETs such as heuristic methods or optimization theory models as shown in [11, 15-20]. Even though data mining solutions have been shown to be efficient and simple to implement in many wireless networking scenarios for detecting faulty/ malicious nodes, these approaches have never been used in the VANET context. Data mining has also the side advantage of making available data that is easily understandable by humans, and that can serve other information extraction purposes. This is why we opted for applying and evaluation a data mining solution in VARM.

Many data mining techniques have been proposed in the past. Recently, in the context of frequent closed itemsets in transaction databases, a number of studies proposed the use of advanced data structures for improving the storage compression of input dataset such as Cats-tree [4], PLT[8] and specially FP-tree[5]. However, FP-tree [5] suffers from a costly storing step. In [8], the authors compared their proposed structure PLT to FP-tree using sparse datasets; they demonstrated that their structure is more efficient than FP-tree structure. We chose to adapt the Itemset-tree concept [1, 2] in our extraction mechanism, VARM, because it is a more compact structure and support independence. Each node in the Itemset-tree is composed of an itemset. This means that the size of a node is not known at priori versus the fixed node size of an FP-tree (24 bytes) and that of Cats-tree and PLT (28 bytes). It is noteworthy that unlike FP-Growth[4] and Closet[5] algorithms, we only consider the lexicographic order as in PLT [8].

2) Itemset-tree Construction Process

Let us consider the transaction database given by Table II. In the Itemset-tree, each node has the following structure : <itemset/support>. Initially, the tree is empty and it is composed by only a root node. We begin by processing the first transaction $m_1m_3m_4m_5m_6$. We derive a node from the root and we add a new node containing the string $m_1m_3m_4m_5m_6$ /1. Next, we process the transaction $m_1m_2m_3m_4m_5$. This transaction and the previous one have in common (or prefixed) the {m_1} item. Hence, the first node is split: we keep the node with m_1/2 and two nodes are derived containing respectively $m_3m_4m_5m_6$ /1 and $m_2m_3m_4m_5$ /1. Processing the third transaction m_2m_4 will leads to the creation of a new node m_2m_4 /1 directly derived from the root node since no items are prefixed in common. The process described below is respected until all the transactions are finished. The result of this process is displayed in Fig.3.

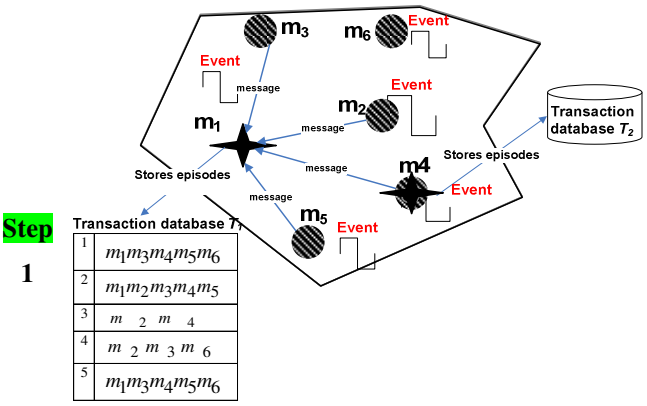

Fig. 2. Network Architecture

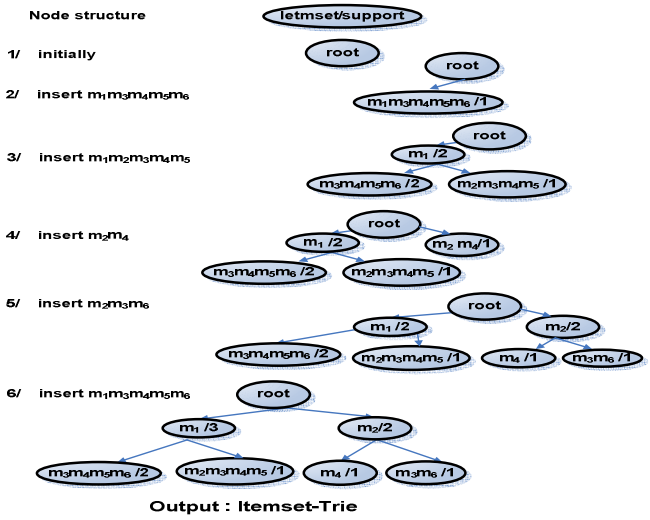

Fig 3. Process of Construction of the Itemset-Tree

TABLE II. THE TRANSACTION DATABASE T1

	TIME SLOT ID	VEHICLES
Step 2	1	$m_1m_3m_4m_5m_6$
	2	$m_1m_2m_3m_4m_5$
	3	m_2m_4
	4	$m_2m_3m_6$
	5	$m_1m_3m_4m_5m_6$

To illustrate this compactness, let us consider the transactions database given by Table II. Fig. 4(a) depicts the associated FP-Tree (12 nodes and 6 levels), while Fig. 4(b) represents the associated Itemset-tree (7 nodes and 3 levels).

a) Closed and No closed Itemset

The generation of closed itemsets and minimal generators (key itemset) are crucial to derive association rules. So, the next example shows how we can find closed and no closed itemset (see Table I for definitions).

The itemset $\{M_1\ M_2\}$ is *not closed* because it does not represent a maximal set of common items with certain objects: all objects with M_1 and M_2 (objects 1, 3, 4, 5) also contain the item M_5 (see table III).

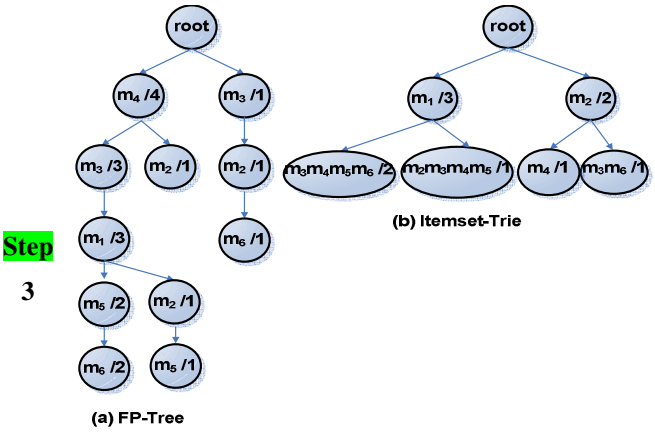

Fig. 4. FP-Tree and Itemset-Tree Associated to the Transaction Database T_1

TABLE III. EXTRACTION CONTEXT K

K	M_1	M_2	M_3	M_4	M_5
1	x	x		x	x
2		x	x		x
3	x	x		x	x
4	x	x	x		x
5	x	x	x	x	x
6		x	x	x	

E. Construction of the Partially Ordered Structures

1) Principles

As output of the second step, we build the Itemset-tree; which can be used to mine closed itemsets and their associated minimal generators repeatedly for different minimum support (*min_supp)* without reconstructing the tree to generate association rules. Since the tree is built representing faithfully the transaction database, the initial Itemset-tree is separated into conditional sub-trees (see Fig. 5 (b), (c)). The same process is done for the conditional sub-trees in order to derive local ordered structures. As example if we fragment the sub-tree in (Fig. 5 (c)), we will have sub-sub-trees.

As we explained before, our main objective is to derive association rules to detect faulty and/or malicious vehicles. For that, we look for constructing ordered structures based on the *precedence relation* to generate association rules; each ordered structure is represented as a graph, in which every node contains the information shown in Fig 6. The process for computing these ordered structures is described in the next section.

2) Example of the Ordered Structures Construction

Let us consider the transaction database given by Table II. The set of the candidate itemsets, with their associated supports, is defined as follows: <m_1/3; m_2/3; m_3/4; m_4/4; m_5/3; m_6/3>. We

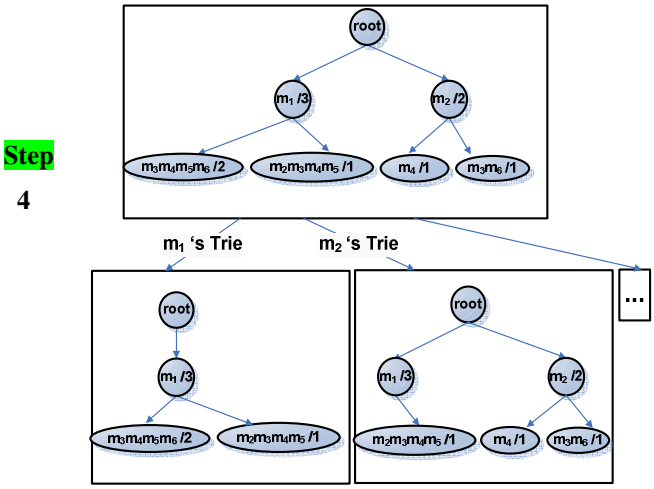

Fig. 5. Main tree and subtrees

will derive six sub structures; each corresponds to an itemset in the list.

a) Construction Process of the Sub-structure Associated with < m_1/3>.

Step 1: Starting with the m_1's conditional Itemset-tree (see Fig. 5.(b)), we can find the associated itemset L_{m1} list :<m_2=1; m_3=3; m_4=3; m_5=3; m_6=2>. From such list we note that the itemsets m_3, m_4 and m_5 are as frequent as the itemset m_1. Hence, they constitute a closed itemset $\{m_1m_3m_4m_5\}$ with a support equal to 3 and with the itemset $\{m_1\}$ as its minimal generator. The itemsets m_3, m_4 and m_5 are removed from L_{m1} and since the latter is not empty, we have to go recursively further in depth and to construct the sub-trees, respectively for the itemsets $\{m_1m_2\}$ and $\{m_1m_6\}$.

Step1.1: From $L_{m_1m_2}$, we discover the closed itemset $\{m_1m_2m_3m_4m_5\}$ with a support equal to 1 and with the itemset $\{m_1m_2\}$ as its minimal generator.

Step1.2: From $L_{m_1m_6}$, we discover the closed itemset $\{m_1m_3m_4m_5m_6\}$ with support equal to 2 and with the itemset $\{m_1m_6\}$ as its minimal generator.

When the treatment of L_{m_1} is finished, and since there are no more elements to handle, we can draw the sub ordered

structure Fig.7 (I). As output, the local Hasse diagram (associated with the m_1's conditional Itemset-tree) can be drawn incrementally. Indeed, the in-depth of the L_{m_1} list enables to connect, first, the closed itemsets $\{m_1m_3m_4m_5\}$ and $\{m_1m_2m_3m_4m_5\}$, and second to connect $\{m_1m_3m_4m_5\}$ and $\{m_1m_3m_4m_5m_6\}$ (Fig. 7 (g, H, I)).

Thus, we can extract the local associated association rules. The Same process is repeated for the other itemsets written in bold in the extraction list from Table II < $m_1/3$; **$m_2/3$; $m_3/4$; $m_4/4$; $m_5/3$; $m_6/3$**>.

Finally, we have all sub-ordered structures and their corresponding local association rules, then we merge the local generic association rules to derive a global one and analyze this new base to detect faulty node and generate useful statistics.

F. Derivation of Generic Bases of Association Rules

Association rules describe how events occur together in the data and are used to represent and identify dependencies between items in the transaction database. The problem of the relevance and usefulness of the extracted association rules is of primary importance. Indeed, in most real life databases, thousands and even millions of high-confidence rules are generated among which many may be redundant. We chose to use association rules because there exist efficient algorithms [1, 4, 5] to mine them from large corpora. In this paper, we are interested in generic association rules defined as follows:

Definition1: Minimal exact association rule: $g \Rightarrow (Y - g)$, where Y is a closed itemset and g is a minimal generator of Y.

Definition2: Minimal approximate association rule: $g \Rightarrow (Y_1 - g)$, g is a minimal generator of a closed itemset Y and Y_1 is a closed itemset, $Y<Y_1$ (Y_1 is a successor of Y) [2].

Fig 6. Node Structure in the Sub Structure Graph

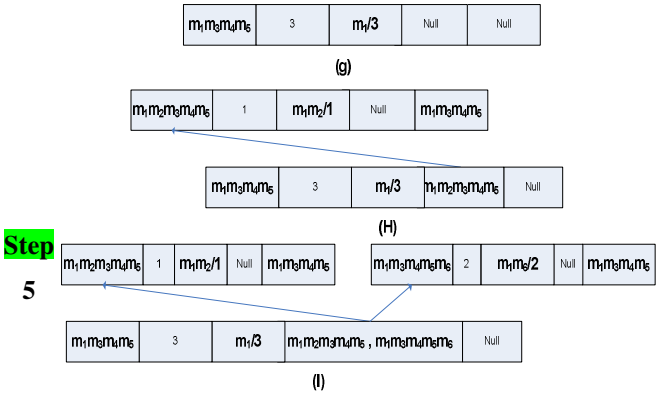

Fig. 7. States of Nodes for the Construction of the Ordered Structure Associated to the 1-itemset m1/3

In the ordered structures generated and shown in Fig.7 (I), each closed itemset is accompanied with its associated list of minimal generators. Indeed, approximate rules represent inter-vehicles relationship deductions, assorted with statistical information, i.e., the confidence, starting from a given vehicle in an ordered structure from a sub-closed-itemset to a super-closed-itemset. Inversely, exact rules are intra-vehicle implications extracted from each vehicle data in the ordered structure. If we apply those definitions, we can extract the following the rules shown in Table IV, computed from the ordered structures generated in Fig. 7 (I).

G. Post Processing of the Results (Association Rules)

The **visualization and post processing** of the association rules is the main part of our approach. In this phase the relevant itemsets should be located in a large collection of potentially interesting itemsets. Consequently, analyzing the results at this stage is essential. Therefore, pertinent decisions may be taken after the extraction of the association rules to re-establish the right relationships.

TABLE IV. LOCAL TEMPORAL ASSOCIATION RULES ASSOCIATED TO FIG. 7 (I)

Minimal exact association rules	Minimal approximate association rules
(1) $m_1 \Rightarrow m_3m_4m_5$, α confidence= $3/3 \Rightarrow$ 100%	(4) $m_1 \Rightarrow m_2m_3m_4m_5$, α , confidence $1/3 \Rightarrow$ 33.3%
(2) $m_1m_2 \Rightarrow m_3m_4m_5$, α confidence= $1/1 \Rightarrow$ 100 %	(5) $m_1 \Rightarrow m_3m_4m_5m_6$, α , confidence= $2/3 \Rightarrow$ 66.6%
(3) $m_1m_6 \Rightarrow m_3m_4m_5$, α , confidence= $2/2 \Rightarrow$ 100 %	

1) Interpretation and Utilization of the Results

From our running example, we extracted exact and approximate association rules (see Table IV).

a) Minimal Exact Rules

From the exact association rule $m_1 \Rightarrow m_3m_4m_5$ (Table IV. (1)), we assume that if we receive an event from vehicle m_1, then, there is 100 percent chance we will receive the same event from vehicles m_3, m_4 *and* m_5 within α units of time. If we do not receive the expected events from one of them, we conclude that the unresponsive vehicle is faulty. This kind of formulation captures the temporal relationships between the corresponding vehicles m_1, m_3, m_4 *and* m_5. The analysis of the association rules allows VARM to identify correlated vehicles that can be used to estimate the values expected from another vehicle, forecast the future sources of events, and detect faulty vehicles.

b) Minimal Approximate Rules

The approximate rules are useful to mine strong approximate dependencies. The idea is that it is interesting to measure not only exact dependencies but also approximate dependencies. In (Table IV. (5)), if an event occurs in m_1 within α unit of time, the expected sources of futures events are m_3, m_4, m_5 *and* m_6 with 66.66% chance. If VARM detects the faulty vehicle correctly, it will react efficiently and rapidly by re-establishing the right relationships.

H. *Predesigned Protection 1: N*

In the technique *1: N*, relationships are reserved in advance. When a failure arises, other relationships which are pre-provisioned are used. Therefore, *1: N* protection, which is similar to *1:1*, except for that one relationship is used to protect *N* relationships. This reactive solution has the advantage to decrease resources demand but introduces a delay. When VARM detects a faulty vehicle, it will proceed to use backup relationships to avoid considering the faulty vehicle. We plan to study the generalization of *1: N* to *K: N,* where *K* protection relationships are used to protect *N* working relationships. Moreover, we will investigate the possibility to use network coding as done in [10].

V. PERFORMANCE STUDY

In this section, we will evaluate the performance of the proposed mining association rules mechanism (VARM). We compare the Itemset_tree, FP_tree[4] and Cats_tree [5] structures in term of compactness when we use different types of datasets (sparse or dense). We chose to compare the last structures using sparse and dense benchmark transaction databases that can be found in [11] to show the efficiency of the structure and the mining algorithm used to extract minimal association rules.

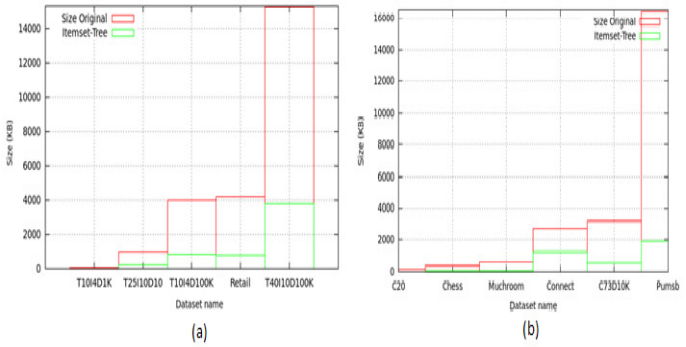

Fig.8. Real Size of the Sparse (a) and Dense (b) Transaction Dataset Versus Itemset-Tree

In Fig. 8-(a), we observed that sparse datasets in average are represented by itemset-tree at most with 24.4% of their original size. While in Fig. 8-(b), dense datasets, in average, are represented at most with 18.8% of their original size. This is very interesting for avoiding technical problems related to itemsets-storage.

In Fig. 9, we chose to evaluate the execution time of the database chess when varying the number of nodes because this database extends on depth and it needs more memory to compute association rules. It is worth noting that in VANETs, the transactions database extracted for 27 vehicles or 40 vehicles is similar to sparse datasets. However, if we prove that the structure is also efficient for dense datasets, it would be interesting as this case might be met with numbers such as 200 vehicles.

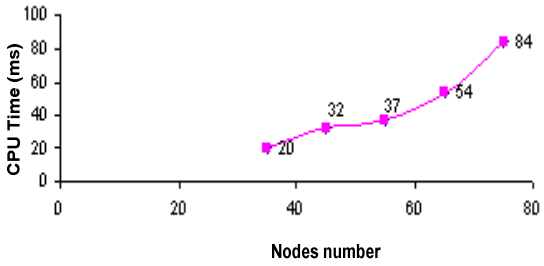

Fig.9. Dense database (chess) / nodes number

Fig. 10. Size of the Itemset-Tree Versus Those of Cats-tree and FP-tree

Fig. 10 shows the performance of the itemset-tree structure compared to the FP-tree and cats-tree in terms of compactness (the original size versus the advanced structure size). We remark that itemset-tree is by far more compact than the other structures proposed in literature.

TABLE V. FRACTION OF SIZE SPARSE/DENSE DATASET AND ITEMSET TREE

Extract context	Original base size in Ko	Avg. num.of transact.	Avg. Num of items in transact.	Itemset-Tree size in Ko	% Approx.
T10I4D1K	38.0	10000	10 nodes	10.00	25%
T25I10D10	947.7	9966	25 nodes	229,7	25%
T10I4D100K	3993.6	10000	10 nodes	839.00	25%
Retail	4198.4	10000	10 nodes	767.81	21%
T40I10D100K	15257.6	100000	25 nodes	3806.10	26%
C20	160	306	35-75 nodes	11,41	7%
Chess	337.4	3196	35-75 nodes	44,83	13%
Mushroom	565	1099	35-75 nodes	48	9%
Connect	2700	17945	35-75 nodes	1180	43%
C73	3174.4	6242	35-75 nodes	512,84	17%
Pumsb	16384	21959	35-75 nodes	1900	24%
Vehicles 27	1	11	2-25 nodes	0	100%
Vehicles 40	1	11	2-37 nodes	0	100%

In table V, we can see how the itemset-tree structure is compact. We observed that with 27 and 40 vehicles network (the last two lines in the table V), this structure is very efficient and suitable. Also, when we compute the exact and approximate association rules, the program is rapidly executed and provides interesting results and pertinent information which leads to identify events correlation and therefore detect faulty and/or malicious vehicles.

The following example presents approximate and exacts associations extracted in a network of 27 vehicles.

1) 3 15 16 17 24 25 26 → 0 1 2 3 4 8 11 12 16 17 20 21 24 25 26 **[3/4]**

2) 3 15 16 17 24 25 26 → 0 1 2 3 4 5 7 8 11 12 16 17 20 21 24 25 26 **[1/4]**

3) 3 15 16 17 24 25 26 → 0 1 2 3 4 6 8 9 10 11 12 13 14 16 17 18 19 20 21 22 23 24 25 26 **[2/4]**

4) 3 15 16 17 24 25 26 → 0 2 3 4 8 11 12 16 17 20 21 24 25 26 [3/4]

5) 3 15 16 17 24 25 26 → 0 1 3 9 10 13 14 16 17 18 19 22 23 24 25 26 [3/4]

6) 3 15 16 17 24 25 26 → 2 3 4 8 11 12 16 17 20 21 24 25 26 [3/4]

7) 3 15 16 17 24 25 26 → 0 3 16 17 24 25 26 [4/4]

The second approximate association rule is not interesting because it appears with confidence equal ¼, which is smaller than the minimum confidence fixed a priori to 50%. However, the association rule number 7 is an exact association rule with confidence equals to 1. Therefore, there is 100 percent chance that these vehicles are correlated. Moreover, we expect that if events happen with vehicles on the left side of a rule, vehicles on the right of the same association rule will be source of events in case of no failure. Otherwise if a vehicle does not generate the expected event, it is then considered either faulty or malicious. In order to distinguish between both cases, we introduce other association rules for this vehicle in order to decide if the vehicle provides false information or is faulty.

VI. Conclusion

In this paper, we proposed a rule-based data mining fault detection technique to detect faulty/malicious vehicles in VANETs based on exchanged routine messages. A side advantage of VARM scheme is that correlated information, displayed via association rules, are easy to understand and subsequently easy to log by humans. In future work, we plan to verify the accuracy of events correlation extraction by a learning technique to learn the historical period.

References

[1] S. BenYahia, Y. Slimani, and J. Rezgui, "A Divide and Conquer approach for deriving partially ordered sub-structures", in Proc. of PAKDD'2005, Hanoi, Vietnam, The Ninth Pacific-Asia, 2005.

[2] J. Rezgui, "Contribution for deriving generic bases of association rules", Master, University of Science in Tunis, Tunisia, Novembre 2004.

[3] N. Pasquier, Y. Bastide, R.Taouil, L. Lakhal, "Efficient Mining of Association Rules Using Closed Itemset Lattices", Information Systems Journal 24, pp. 25–46, 1999.

[4] J. Han, J. Pei, Y. Yin, "Mining frequent patterns without candidate generation", in Proc. of the ACM-SIGMOD Intl. Conference on Management of Data, Dallas, Texas, pp.1–12, 2000.

[5] W. Cheung, O. Zaiane, "Incremental mining of frequent patterns without candidate generation or support constraint", in Proc. of the Seventh International Database Engineering and Applications Symposium (IDEAS), Hong Kong, China, 2003.

[6] G. Grahne, J. Zhu, "Efficiently using prefix-trees in mining frequent itemsets", in Proc. of the Workshop on Frequent Itemset Mining Implementations (FIMI), Florida, USA, 2003.

[7] K.K. Loo, I. Tong, B. Kao, and D. Chenung, "Online Algorithms for Mining Inter-Stream Associations from Large Sensor Networks", in Proc. of PAKDD'2005, Hanoi, Vietnam, The Ninth Pacific-Asia, 2005.

[8] A. Bouckerche and S. Samarah, "A novel algorithm for mining Associations rules in Wireless Ad Hoc Sensor Networks", Parallel and Distributed Systems, IEEE Transactions on, july 2008.

[9] S. Hakami, et al., "Detection and Identification of Anomalies in Wireless Mesh Networks Using Principal Component Analysis (PCA) ", in Proc. of Parallel Architectures, Algorithms, and Networks, pp. 266-271, 2008.

[10] Al-Kofahi, O.M. and A.E. Kamal, "Network Coding-Based Protection of Many-to-One Flow Networks", in Proc. of IEEE Mobile Adhoc and Sensor Systems (MASS), pp. 1-10, 2007.

[11] R. Lin, E. Khalastchi and G. A. Kaminka, "Detecting anomalies in unmanned vehicles using the Mahalanobis distance", in Proc. of ICRA, pp. 3038-3044, 2010.

[12] J. Rezgui, S. Cherkaoui, and O. Chakroun, "Deterministic Access for DSRC/80211.p Vehicular Safety Communication", in Proc. of The 7th International Wireless Communications and Mobile Computing Conference (IWCMC 2011), 2011.

[13] D. Jiang, et al., "Design of 5.9 ghz dsrc-based vehicular safety communication", Wireless Communications, IEEE, Vol. 13, pp. 36-43, 2006.

[14] H. Ching-Ling, et al., "Adaptive intervehicle communication control for cooperative safety systems", Network, IEEE, Vol. 24, pp. 6-13, 2010.

[15] T. Zhou, et al., "Privacy-Preserving Detection of Sybil Attacks in Vehicular Ad Hoc Networks", in Proc. of the Fourth Annual International Conference on Mobile and Ubiquitous Systems: Networking Services (MobiQuitous), pp.1-8, 2007.

[16] M. E. Zarki, et al., "Security issues in a future vehicular network", in Proc. of EuroWireless, 2002.

[17] M. Raya and J.-P. Hubaux, "The security of vehicular ad hoc net-works", in Proc. of SASN, 2005.

[18] K. Sampigethaya, et al., "Caravan: Providing location privacy for vanet", in Proc. of ESCARworkshop, 2005.

[19] J. Y. Choi, M. Jakobsson, and S. Wetzel, "Balancing auditability and privacy in vehicular networks", in Proc. of Q2SWinet, 2005.

[20] E. Coronado, S. Cherkaoui,"A secure service architecture to support wireless vehicular networks", Special Issue on "Security, Trust, and Privacy in DTN and Vehicular Communications", International Journal of Autonomous and Adaptative Communications Systems (IJAACS), Inderscience, Vol3, No.2, pp 136-158, 2010.

[21] IEEE Vehicular Technology Society: 5.9 GHz Dedicated Short Range Communications (DSRC):. http://grouper.ieee.org.groups/scc32/dsrc