Tracking Spoofed Locations in Crowdsourced Vehicular Applications

Lautaro Dolberg, Jérôme François, Thomas Engel

Tracking Spoofed Locations in Crowdsourced Vehicular Applications

Lautaro Dolberg, Jérôme François, Thomas Engel
University of Luxembourg
Email: firstname.lastname@uni.lu

***Abstract*—Position information is essential for many vehicular applications like traffic information and route planning but also for enabling vital network routing services. However, the accuracy of the latter is highly dependent of a precise location service which might be altered by attackers forging falsified position data. In this paper, we propose a new method for tracking and removing location spoofing anomalies in large scale position based services assuming no control of the communication infrastructure or additional hardware or software at the client side. Our system uses aggregation of location information combined with relevant stability metrics.**

I. Introduction

Positioning systems are considered as a core component in vehicular applications. Firstly, specific routing protocols, qualified as Vehicle-to-Vehicle and Vehicle-to-Infrastructure, may rely on vehicle positions to infer the forwarding path of packets [1], [2], [3]. In [4], the authors propose to rely on vehicle movements to deliver packets. In parallel, we are being witness of a fast growing application market for improving driving experience like road traffic aware trip planning, alert broadcasting or car-pooling services based on drivers' habit. Most of these applications rely on a combination of human cooperation and technological improvements specially brought by the high penetration of mobile devices like smartphones equipped with positioning services such as GPS. Therefore, this enables vehicular-based applications without specific investment or infrastructure deployment like roadside units or embedded on-board units.

The infrastructure is the communication channels between the users and the service. Authentication does not prevent attack in crowd-sourced app since the subscription is usually open and so creating numerous sybils is easy. This also makes blacklisting ineffective. Injecting numerous cars leads to have some of them at plausible positions which limits the detection on such observations.

The services mentioned above require a very reliable database of vehicle positions. Those databases can be poisoned with forged positions reported by attackers (location spoofing attack). Thus, an attacker can fake traffic congestion or alter traffic routing systems [5]. While most of popular vehicle based applications like traffic congestion avoidance or event network level routing [6] consider aggregated data, this kind of attacks are still valid by reporting a large number of spoofed or faked vehicle positions.

In this paper, we propose a method for position spoofing detection. It is inspired from works relying on movement plausibility verifications [7], [8], [9] but enhances the scalability by leveraging aggregation which so avoids the costly verification of the movements of each individual vehicle. In fact, vehicles are grouped within different areas of various sizes of a map depending on the traffic density using Multidimiensional Aggregation Monitoring (MaM) [10]. As a first contribution, we designed dedicated metrics that are applied to such aggregated representations to verify the global vehicle repartition on a map and the associated movement dynamic. Unlike standard approaches relying on individual car movements, our method focuses large datasets such as those which can be collected from navigation systems and crowd-sourced applications. The second contribution is a recovery mechanism to safely remove falsified data.

This paper is structured as follows. Section II introduces the problem and related work. Section III describes our location spoofing detection and recovery method. Section IV details the datasets used in the evaluation in Section V. Finally, Section VI concludes the paper and outlines future work.

II. Location Spoofing

A. Context

In the past, reporting services were reserved to few devices installed in specific vehicles, in particular for logistic. The use of mobile devices with embedded GPS has been growing for the last decade, and with its expansion, more and more devices are capable to perform position reporting. That is why location aware services have appeared in smart-phone application markets and in particular crowd-sourced services. Nowadays, there are many private and public initiatives in this context for road traffic monitoring like Google Maps [1], INRIX [2] or LuxTraffic [3].

Therefore, crowd-sourced mobile application is the main scenario considered and illustrated in figure 1 where everybody can freely report his positions to a central service which then infers traffic conditions in real time. Our approach is not dependent on the application and is useful for all applications that require location information. As illustrated in the figure, the service provider does not control the infrastructure and the service is simply available through Internet.

[1] http://maps.google.com [11] – accessed on August 7th 2013
[2] http://www.inrix.com – accessed on August 7th 2013
[3] http://www.luxtraffic.lu/ – accessed on August 7th 2013

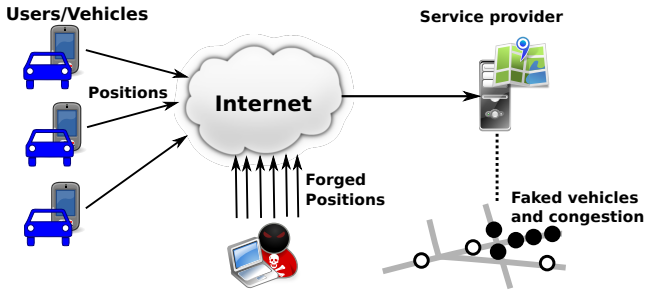


Fig. 1: Location spoofing – traffic monitoring use case

B. Problem Description

As a crowd-sourced application, there is no strong restriction on users, everybody is basically identified with pseudonyms. Thus, an attacker can inject such erroneous data, as shown in Figure 1, using multiple vehicle identities (sybil attack) using a standard computer which is be considered as the most realistic and threatening scenario [12]. Complementary approaches to automatically identify such unusual or unauthorized devices also exist (device fingerprinting) but are dependent on the specific protocol used [13], [14]. Even without a success rate of 100%, an attacker may cause unforeseen and catastrophic consequences to location based services (creating traffic congestion and so indirectly rerouting people to specific locations, indicating false collisions which may provoke unnecessary braking, etc).

In this paper, we aim to identify and localize an attacker who submits forged positions of numerous vehicles to a position-based service. Obviously, injection may be done continuously to be more stealthy and effective. Hence an attacker can mimic a driven vehicle by reporting multiple positions along a path in a coherent timely manner.

In particular, the following problems are addressed in this paper:

- Location spoofing detection: the goal is to detect that an anomaly occurs in the reported locations.
- Spoofed position recovery: assuming that the location spoofing has been detected, the goal of the recovery consists into precisely identifying which positions are subject to the attack for discarding the corresponding records in order to keep the database of reported positions (and so the road traffic flows) in a safe state.

While the first objective is to detect when a spoofing attack occurs, the second one tries to localize where it occurs.

C. Related work

A risk analysis of roadside attackers is provided in [12] highlighting location spoofing attacks as the major vulnerability of VANETs (Vehicular Ad-hoc NETworks) without providing any counter-measures. To verify locations, there are several approaches relying on the infrastructure to check the coherency between the claimed location and the real existence of the vehicle in this location. For example, the distance between a fixed probing station can be estimated using challenge response time [15], [16]. The deployment of road side units is also a pre-requiste in [17], [18]. Assuming our context, such approaches are not viable since the service provider has no control of the infrastructure.

Vehicles may also be equipped with additional sensors to check the location of their close neighborhoods [19] which requires a vehicle modification contrary to the recent trend is to use already connected devices like smartphones. Authenticating users properly has also been proposed [20], [21], [22], [23] but this assumes a defined and trusted set of users which is antagonist with crow-sourced applications which allows the attacker to create large volume of sybils. This is also different to scenarios assuming few misbehaving vehicles while a key management infrastructure is used [24], [25].

Plausibility checks consist in verifying a claimed position regarding physics properties. In [26], this is done by estimating the position based on the radio signal strength. In [7], a node (a vehicle) checks itself that information it receives from another is coherent regarding the radio range, the map topology, the speed calculated between two consecutive beacons, the number of vehicle at the same location, etc. The authors extend this approach by enabling cooperation between nodes [8] similarly to [27] which used an ellipse based location estimation. Such an approach is also presented in [28] where each node constructs and shares its location representation of the VANET. Time-of-flight of signals between two cooperative nodes is leveraged in [29]. More specifically, trust management for the vehicular context is discussed in [30]. In these different works, radio frequency measures like signal strength or timing differences are the major metrics. In [31], the nodes report the detected misbehaviors to a centralized service that combines them in order to track, and so to exclude, the attackers at a long term perspective. Even in that case, location spoofing detection relies on end user devices while our approach shifts the location spoofing detection at the service provider side. In such a way, resources are saved from the client side and does not constrain the client devices.

However, our technique also leverages plausibility checks by only seeking for incoherency in the reported positions. While a naive approach could check if there several vehicles at the same position, the authors in [9] show that it has to consider the velocity and multiple measures for a single vehicle due to the errors induced by the positioning systems and unknown exact vehicle sizes. This makes such a verification complex when applied to thousands of vehicles and does not prevent an attacker to inject faked vehicles at viable positions even if a certain proportion of them are overlapping existing ones. Hence, this paper proposes to apply plausibility checks on aggregated data in order to avoid the costly analysis of each single vehicle position.

III. Aggregation Based Detection

A. System Description

In the upper part of Figure 2, a position reporting log is obtained and contains samples X,Y coordinates (longitude, latitude) for vehicles during a given period of time.

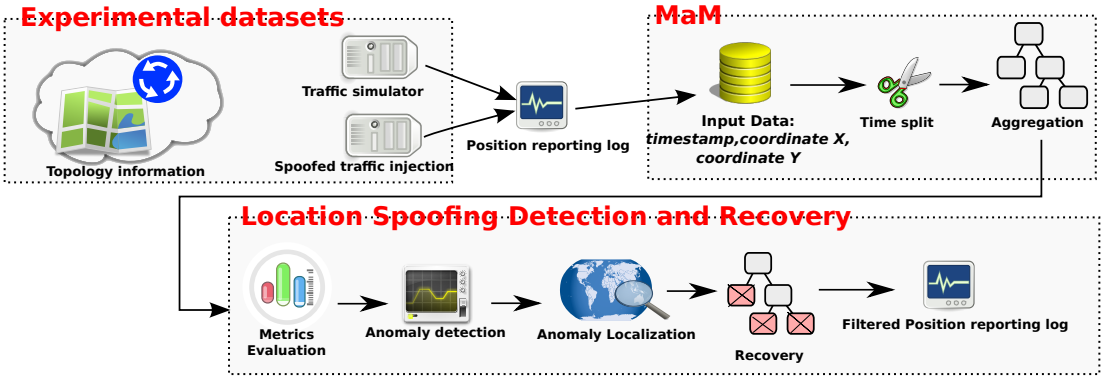


Fig. 2: System overview

A key component of our system relies on *MaM* (Multi-dimensional Aggregation Monitoring) [10], this component is responsible for splitting and aggregating large multi-dimension data collections (in our context, vehicle positions) into trees. MaM-based aggregation shrinks very large data collections and so improves the scalability but MaM only outputs a new representation of data. Hence, this paper proposes metrics and methods for analyzing such outputs.

In particular, a stability analysis over a sequence of trees helps in establishing an experimental threshold to find nodes (corresponding to geographical areas) under attack. Once malicious nodes are identified, malicious data can be discarded from the trees to sanitize data.

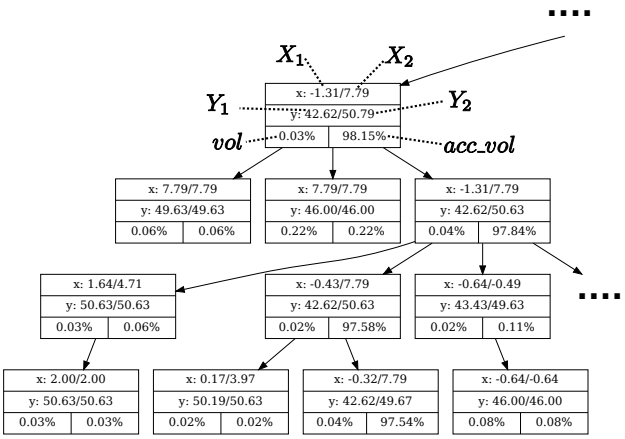


Fig. 3: *MaM* sample output (Due to visualization, values are rounded with two decimal places)

B. MaM

Multidimensional Aggregation Monitoring (*MaM*)[10] aggregates large collections of data by (1) dividing data by intervals of β seconds and (2) creating an aggregate tree of data for each time window. This tool inspired from single dimension aggregation [32] was used to aggregate IP addresses or domain names for detecting anomalies [33]. Data structures used for aggregation are modified Tries [34] relying data's natural hierarchy. For example, the IP address hierarchy is the standard subnetwork division.

MaM performs temporal aggregation over one or several features. For example, it is possible to aggregate source and destination IP addresses meanwhile. The advantage is that *MaM* automatically splits the feature space (no need to fix the granularity) and does not need any predefined order like a first aggregation on destination IP subnetworks and then on source IP subnetworks. In fact, aggregation is guided by the events to monitor, *i.e.* representing a minimal percentage of the activity. This is defined by the parameter α. While details about MaM data structure and algorithms are out of the scope of this paper, the intuition behind is that data are incrementally clustered regarding the density of data in the multi-dimensional space with different levels (like "small" and "large" rectangles in 2 dimensions) and clusters are hierarchically linked (like a rectangle embedded in a larger one).

C. MaM for vehicle locations

In this work, we decided to model an area as bounded intervals. In fact, an area is defined as bi dimensional space using Cartesian coordinates to represent the longitude and latitude (altitude is not considered). Hence, the areas are represented as rectangles defined by two points: $((X_1, Y_1) : (X_2, Y_2))$. The most specific representation is an interval of a single point: $X_1 = X_2$ and $Y_1 = Y_2$.

In the sample output shown in Figure 3, each node represents a specific rectangular area defined as:

1) X dimension range: (X_1, X_2) where $X_1 \in \Re$ and $X_2 \in \Re$
2) Y dimension range: (Y_1, Y_2) where $Y_1 \in \Re$ and $Y_2 \in \Re$
3) Percentage of vehicles in the corresponding area excluding its children: $vol \in \Re$
4) Cumulated percentage of vehicles in the corresponding area (including its children): $acc\_vol \in \Re$

The corresponding area of a node is actually embedded into the area associated to its parent node.

As highlighted before, a hierarchical model has to be built over such dimensions. In order to keep the advantages of *MaM*, the areas are not fixed in advance as we could have done using a grid-based approach. Rectan-

gle are built regarding the percentage of activity which, in this case, represents the percentage of vehicles. To do so, the areas are created on the fly by assembling points together. For example, assuming two individual points as $((X_1, Y_1) : (X_1, Y_1))$ and $((X_2, Y_2) : (X_2, Y_2))$, this will entails the creation of the area: $((mix(X_1, X_2), mix(Y_1, Y_2)) : (max(X_1, X_2), max(Y_1, Y_2)))$. When the third point has to be added, either it falls into this created area which implies the creation of an embedded area, or it falls outside creating a parent area. This process continues until new data points have to be added. Hence, this automatically creates a hierarchy between areas which is not known a priori unlike IP subnetworks. In the meantime, each area maintains a counter about the number of vehicles it contains. Aggregation is then performed from the leaves to the root. For each node, if $vol > \alpha$, the node is kept, otherwise it is discarded and its vol value is added to the one of its parent node. This aggregation process is the standard one of MaM which is fully described in [10].

D. Metrics

We propose a method for analyzing series of aggregated trees based on stability over time. The intuitive idea behind is similar to plausibility check by considering that vehicles movements from one area to another are bounded by physical properties but also depending on the traffic conditions. To achieve that, we defined a stability metric that captures the dynamic of the traffic, i.e. the movement of the car, by comparing consecutive vehicle positions. Observing the variation of the stability when no attack occurs, we are then capable of detecting abnormal variations. The aggregation helps in considering the global road traffic dynamic. In addition, this avoids to detect isolated deviant behaviors which are not necessary malicious and, in fact, does not have an impact on the large scale location-based service that relies on mass information.

As explained, the stability captures the dynamic of the car. It is a bounded value between 0 and 1. Assuming cars blocked in a traffic jam, the stability will be very high. On the contrary, cars moving very fast entails a low stability. Therefore, the goal is to evaluate the stability of the traffic (and so vehicles positions) between consecutive time windows. To do so, it is firstly required to define the stability between two nodes, $n1$ and $n2$. It is higher if the distance between the associated areas ($Area\_Distance(n1, n2)$) are close, if the overlap between them is high ($Area\_Com(n1, n2)$) and if the number of vehicles is similar:

$$\begin{aligned} Stability(n1, n2) = \eta \times Area\_Com(n1, n2) + \\ \psi \times Area\_Distance(n1, n2) + \\ \gamma \times (1 - |n1.vol - n2.vol|) \end{aligned} \quad (1)$$

The common space is evaluated regarding the ratio between the intersected rectangular area and the merged rectangular area: $Area\_Com(n1, n2) = \frac{\cap(n1,n2)}{\cup(n1,n2)}$

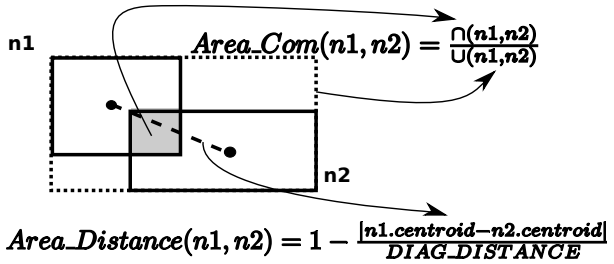


Fig. 4: Area comparison

The distance is evaluated regarding the centroid of the areas and normalized regarding the diagonal distance of the monitored map: $Area\_Distance(n1, n2) = 1 - \frac{|n1.centroid - n2.centroid|}{DIAG\_DISTANCE}$

The common area and centroids can be easily retrieved from the coordinates and Figure 4 illustrates them. Evidently, areas are not always intersected and in this case, $Area\_Com(n1, n2) = 0$. Without specific knowledge, $\eta = \psi = \gamma = 1/3$ which weights equally each factor.

The objective is to compare the stability between t_i and a previous tree t_j. To achieve that, the individual stability of each node $n_i \in t_i$ is calculated. This is calculated by comparing with the most similar node in t_j which is $mostsim(n_i, t_j) \in t_j$ such that $\forall_{n_j \in t_j} Stability(n_i, mostsim(n_i, t_j)) \geq Stability(n_i, n_j)$.

Finally, to globally assess the abnormality of a tree t_i, the average stability over all nodes is considered:

$$Avg\_stab(t_i, t_j) = \frac{\sum_{n_i \in t_i} Stability(n_i, mostsim(n_i, t_j))}{|t_i|} \quad (2)$$

E. Algorithms

In this section, the core algorithms involved in the location spoofing detection and recovery are presented. As mentioned before, the location spoofing attack is detected by evaluating the global stability in equation (2) and detecting abnormal variation. For sake of clarity, we use a single profile which corresponds to set a threshold value of an acceptable stability variation. It is also possible to imagine more complex models taking in account peak hours for example. However, the method remains the same by using the stability measure to establish multiple profiles depending on certain criteria. To not be biased by local events, our approach promotes the use of a sliding window by comparing a tree with its S predecessors.

Hence, the algorithm 1 computes the stability of all nodes of a tree assuming a list of predecessors. The idea is to compare pairwise nodes from the current tree (noted as parameter t) against the previous trees within the sliding window. This can be observed in line 6 where the most similar node (having the highest stability) of each node of t is searched in the predecessors, as defined in the previous section. Then, the average stability for every single node t is computed over all the predecessors (line 7). In line 3, the algorithm iterates over

each node of the input tree which leads to $O(n)$ assuming n nodes in a tree. For each node a look up over all S predecessors is performed in line 5, and in line 6 the most similar node is searched for every element on the predecessors list, with so a final cost of $S \times n$. In line 7 scalar operation are performed with a constant cost. Thus, the average complexity of algorithm 1 is $O(n^2)$ where n is the number of nodes since S is fixed constant.

Algorithm 1 Calculate the stability for the nodes of a given tree and its predecessors

Input: $t : Tree$
$predecessors : List < Tree >$
Output: $StabMap(t, predecessors) : Map < Node >< Float >$
1: $nodes \leftarrow$ nodes from t
2: $res \leftarrow Map()$
3: **for** n in $nodes$ **do**
4: $st \leftarrow 0$
5: **for** $t_i \in predecessors$ **do**
6: $c \leftarrow mostsim(n, t_i)$
7: $st \leftarrow st + Stability(n, c)$
8: **end for**
9: $res(n) \leftarrow st/lenght(predecessors)$
10: **end for**
return res

The detection and recovery algorithm 2 works as follows. If the average stability variation equation (2) exceeds the threshold θ, then an attack is detected as shown in line 2 which requires to compute the stability of all nodes of the current tree based in line 1 which executes algorithm 1. The function $values$ allows to extract only the values of the map data structure. In fact, the recovery is an iteration process which removes one by one the least stable leaves until the difference between the current and past average stability values does not exceed the threshold θ as shown in lines 4-6.

The computation of the stability of the nodes is done once in line 1 which has a complexity of $O(n^2)$ where n is the number of nodes in a tree. The average stability is calculated bu iterating over all values in $stabMap$ which corresponds to the number of nodes. In parallel, the least stable leaf can be found, which means that operations in lines 2 - 4 are executed meanwhile by iterating over all nodes. In the worst case (in case every node needs recovery), the while loop condition will be also evaluated n times. This finally leads to a complexity equals $O(n^2 + n^2) = O(n^2)$.

Therefore, the general process takes as input position data with timestamps in order to create aggregated trees on the fly depending on the time window size β and the aggregation threshold α. When a time window ends, the corresponding tree is preprocessed using the algorithm 1 before the detection and the recovery is triggered in algorithm 2.

Algorithm 2 Detection and Recovery

Input: $current : Tree$
Input: $predecessors : List < Tree >$
Input: $previous\_stab : float$
Output: $Recovery(current) : Tree$
1: $stabMap \leftarrow StabMap(current,\ predecessors)$
2: **while** $|avg(values(stabMap)) - previous\_stab| > \theta$ **do**
3: $leafs \leftarrow$ leafs from $current$
4: $least\_stable \leftarrow l \in leafs, \forall l' \in leafs, stabMap(l') \geq stabMap(l)$
5: remove l from current
6: remove l from $stabMap$
7: **end while**
return $current$

IV. Data Set

A. Traffic Simulation

In order to perform realistic experiments, the simulator SUMO (Simulation of Urban MObility) [35] was used in order to generate vehicle traffic flows in an urban environment.

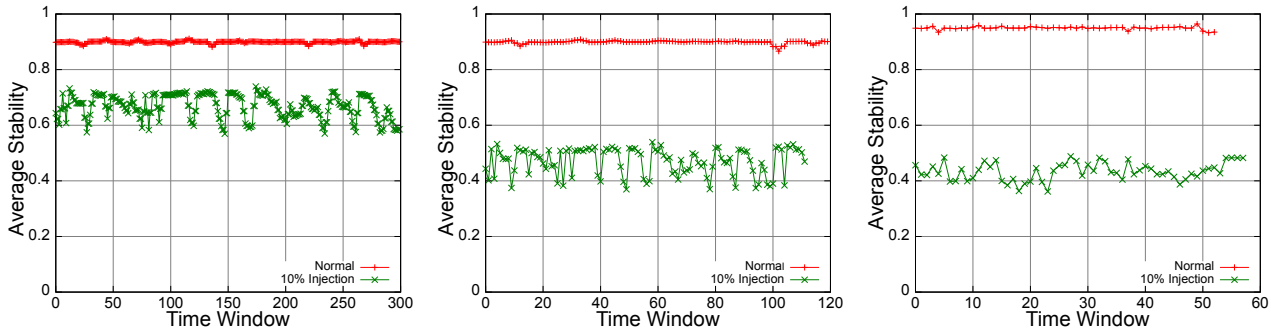
B. Urban Scenarios

Our evaluation is based on two scenarios to represent different possible urban topologies. A Manhattan mobility model relies on a grid road topology and is representative of modern cities especially in North America. In order to have a good trade-off between SUMO performances and representativeness, the grid road topology is bounded to a 5 km $\times$ 5 km square containing regular 100 meters length blocks. Because such a topology is not representative of old European cities. Hence, we included the city of Luxembourg as a complementary approach for simulating traffic. For fairness, the Luxembourg scenario is also bounded to a 5 km $\times$ 5 km which is centered in the city center. This is obtained using Open Street Map[4].

C. Traffic Simulation

For simulating traffic in both scenarios, a random mobility pattern is used. It consists in defining, for each normal vehicle, a random departure and an arrival street point. In addition, when the car arrives at its destination, its location information is not provided anymore to mimic a real vehicle. For both scenarios we generated traces for 10 consecutive hours. A total of normal 220,000 vehicles were injected along the simulation. The number of misbehaved vehicles (spoofed locations) is then dependent on the simulations as explained in next sections. For being realistic, the cars are not injected simulateously but also at a random number, every seconds. Around 20833 new cars are injected every hour as highlighted in Table I, which gives also some other characteristics of the datasets. Moreover, each vehicle reports its position every second.

[4]http://www.openstreetmap.org/



(a) β = 120 seconds (b) β = 300 seconds (c) β = 600 seconds

Fig. 5: Average Stability for 10 hours simulation on a Manhattan Grid without attack or with 10% of injected data

	Manhattan	Luxembourg
Total Cars	220,000	220,000
New cars per hour	20833.33	20833.33
Average Car per Block	8	11
Average Speed	15	17
Average Car Trip Time	12 min	8 min

TABLE I: Simulation Performance in numbers

D. Malicious Traffic Data Generation & Injection

Malicious traffic was injected in order to extend each original dataset during our experiments. The main principle consists into injecting faked reported positions of vehicles in a predefined area which thus mimics spoofed vehicles in this area. Therefore, the following parameters are used:

- Time: the time interval to perform the data generation.
- Center: the center of the area where faked vehicles are injected
- Radius: the radius size of the area where faked vehicles are injected
- Volume of cars: overall amount of injected vehicles.
- Frequency: proportion of vehicles per time unit to be injected regarding the number of normal cars.

The injection takes in consideration the map topology for creating vehicles at valid locations. Besides, the traffic injection models may generate spoofed location attacks against multiple areas. To achieve that, several locations (center and radius) can be defined. In such a case, the volume of cars and the frequency of data generation are global to all targeted locations. In our experiments, the radius is fixed to 500 meters the number of targeted locations is 2 for the Luxembourg scenarios and 3 for the Manhattan scenario. We deliberately choose small values in order to strengthen our approach by considering targeted attacks unlike global attacks that could impact all the map and so be highly visible.

As highlighted in section II, a unique identification of every vehicle would be helpful but does not sound plausible as we are focusing on open crowd-sourced applications relying on a non dedicated infrastructure. This eases the task of the attacker to create faked identity as many as he desires. In addition, drivers and so vehicle identification is a major privacy issue. We implemented the most general model without identifying vehicles which thus conforms to our detection approach.

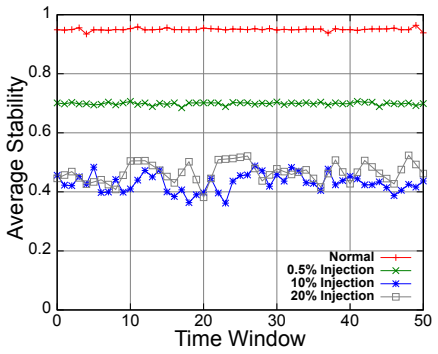
V. Experimental Results

In this section, experimental results are presented. The first experiment was conducted to evaluate the impact of the attacks on the stability and so to show that a threshold-based technique (θ) is viable. This experience would allow us to calibrate β parameter (aggregation time window). The second experiment assesses the impact of the window size. The third experiment is dedicated to the localization of the attack and its recovery. Then, this section also highlights the gain of using the aggregated trees compared to individual vehicle positions.

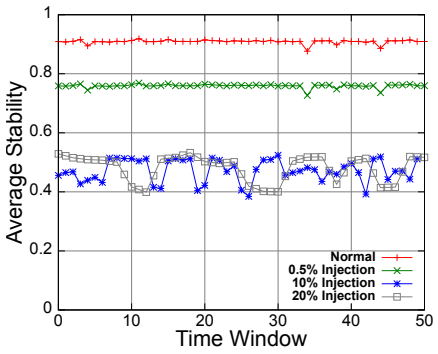
Preliminary experiments help in identifying good parameter values for the aggregation threshold $\alpha = 2\%$ and the sliding window size $S = 5$ for stability computation.

A. Window size impact

In order to characterize regular traffic stability, Figure 5 compares the average stability when there is no attack and when 10% of faked vehicles are injected. As mentioned before in this experiment we aim to asses the impact of β (time aggregation window). This experiment considers a Manhattan grid topology and the window size is successively fixed to 120, 300 or 600 seconds. The average stability shown in Figure 5 is computed over ten independent simulations. Regardless the window size, the average stability when an attack occurs is quite different and so an attack could be easily detected. Next section considers more stealthy attack with less faked vehicles. However, as illustrated in the figure 5 and especially in figure 5(c), increasing the window size improves the separability of the curves which thus will help in detecting the attack. In fact, this discards local bias by merging much position data. However, this will delay the attack detection until the time window ends. Assuming a standard context of road traffic monitoring, 600 or 300 seconds seems reasonable and provides good results in Figures 5(b) and 5(c). Therefore, such values for β are used in the other experiments.



(a) Manhattan topology



(b) Luxembourg topology

Fig. 6: Average Stability with various attack level, β = 600 seconds

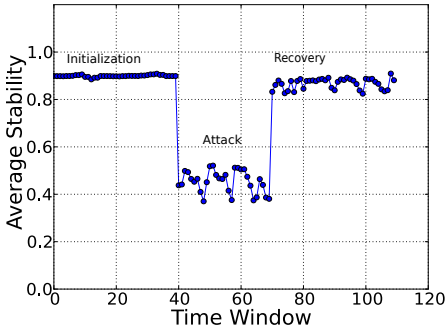


Fig. 7: Attack injection with recovery phase, β = 300 seconds, Manhattan topology

B. Attack Identification

In this experiment, $\beta = 600$ seconds and the objective is to assess the impact of the attack injection level, between 0.5% and 20% of normal vehicles. In Figure 6, both the Luxembourg and the Manhattan scenarios are considered. The curve on top of each graph represents the average stability as defined in equation (2) when no attack occurs. In such a case values are between 0.85 and 0.95. Logically, when the attack aggressiveness increases, the average stability drops in higher proportion. However, after reaching 10% of spoofed vehicles, reaching a limit of disturbance for this experiments. On a tiny scale with only 0.5% injection (the number of vehicles injected correspond to 0.5% of total volume of cars), the average stability is lowered around 0.7 and so clearly distinguishable from normal ones. Therefore, setting $\theta = 0.15$ in algorithm 2 is enough to detect stealthy attacks independently of the topology.

C. Recovery

Previous experiments show that establishing a threshold experimentally is viable. It can be easily determined through learning and past observations of the stability when no attack occurs. Assuming that an attack is detected thanks to this method, this section assesses the ability of our system to recover to a safe state by sanitizing the data from spoofed reported locations. In algorithm 2, this corresponds to remove unstable leaf nodes. Using $\theta = 0.15$ as previously established, Figure 7 depicts the average stability before the attack, during the attack and when the recovery is performed. When the attack is launched starting from time window 40. Until the time window 75 the attack causes a massive drop of average stability and also produces a great oscillation. In the last phase of the sequence, recovery phase is launched. Despite the attack continues, by removing nodes responsible for the average stability decay, the stability is restored to acceptable values. In reality, recovery phase should start earlier (when the attack is detected) and we voluntary delayed it in this experiment to highlight the changes in the stability value.

While restoring an acceptable stability is quite easy by removing leaves until the difference between the past observed stability is under the variation threshold θ, it is important to evaluate if the discarded data properly belongs to the attack (true positives) or not (false positives). This is evaluated in Figure 8 by calculating the True Positive Rate (TPR) and False Positive Rate (FPR). FP are due to random normal traffic which, at some high attack rates, can be interpreted as an alert.However, most of them occur with excessively high attack rates. We'll also evaluate the impact of theta and including other metrics like the specificity.

The FP represents the real road traffic that will be discarded and so can have an hight impact on monitoring, undetected traffic jam for example. Such examples can be more discussed to show how FP may impact applications. Where we assess a "single" attack, it corresponds to test every individual one separately, and then compute an average, but all are tested. These rates are calculated based on the stability value of the leaf nodes. For instance, assuming $\theta = 0.15$ and a normal stability around 0.9 as highlighted in the first phase of figure 7, this means that we are considering an absolute threshold of 0.75. In such a case, the TPR reaches 89% with less than 27% of false positives. FPR can be drastically reduced using a smaller threshold like 0.4 which leads to a TPR equal 75% and FPR equal 4%.

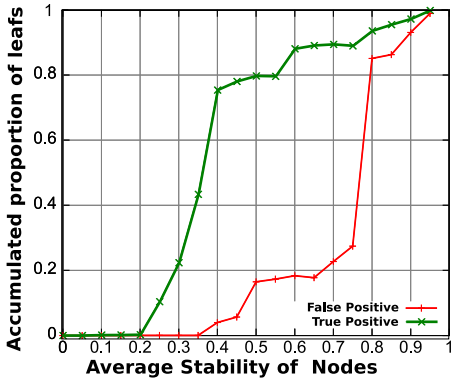


Fig. 8: Threshold detection for simulation containing attacks with up to 10% of increased traffic

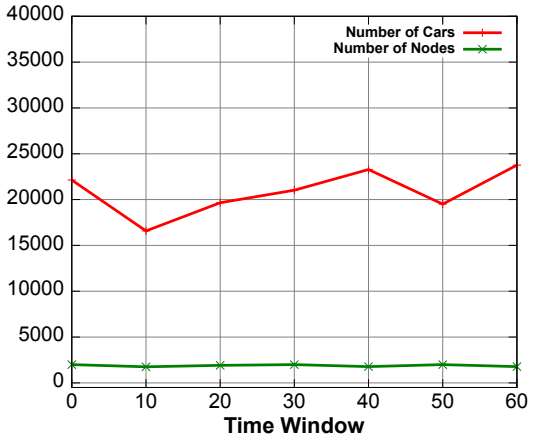


Fig. 9: Number of cars and size of the trees, 10 hours simulation, β = 600 seconds

D. Performances

As discussed in section II-C, a simple approach could verify if two vehicles are at the same position but this needs further refinement to be viable. However, considering that such a case is possible, this entails to verify pair-wise vehicles. Hence, with V vehicles, $\frac{V(V-1)}{2}$ verifications have to be done. For computing the stability of all nodes of a tree, detecting anomalies and recovering, it has been shown that the complexity is $O(n^2)$ in section III-E, assuming n as the number of nodes in a tree. From a theoretical complexity point of view, this is similar to consider complexity assuming individual vehicles (quadratic complexity in both cases) but, in practice, the tree size is highly lower than the number of vehicles due to the aggregation process. Especially, Figure 9 highlights the number of individual vehicles and the size of the trees. The latter is at least 1760 and varies around 2000. In contrast, the number of cars is always around 20,000 which leads to around 200 millions comparisons. This is ten times more than using our tree-based approach when $S = 5$ (5×2000^2). In addition, our process also includes the recovery mechanism and so is not only limited to the detection.

VI. Conclusion

This paper describes a new location spoofing detection and recovery mechanism that can be applied to a large scale database of position data such as those used in many vehicular applications. Our approach leverages MaM for collecting and aggregating data. On top of it, we defined dedicated metrics and associated algorithms to compute them to verify plausible global vehicle movements. In this work we were able to show that average stability metric is a rough indicator of anomaly presence in traffic flows which have been realistically simulated. The main advantages of the aggregation are the global evaluation the traffic without considering too small events and the scalability improvement. In addition, a recovery mechanism is proposed which allows to safely removed most of the anomalous data while discarding very few benign data. As a future work, we would move towards testing our methods with other datasets in particular real traffic traces as well as including other features in the multidimensional aggregated trees like the semantic of vehicle journeys. As a future work we are already working on a realistic attack generator based on real data collected through location based services. However, for data collection to be representative we face a scale challenge. Moreover, to re validate our experiments in specific topologies would not add more generality. Different kinds of architectural design of cities are already taken in account for a future work.

Acknowledgement: this work was partially funded by IoT6, a European FP7 funded project under the grant agreement 288445 and MOVE, a CORE project funded by FNR in Luxembourg.

References

[1] L.-D. Chou, J.-Y. Yang, Y.-C. Hsieh, D.-C. Chang, and C.-F. Tung, "Intersection-based routing protocol for vanets," *Wirel. Pers. Commun.*, vol. 60, no. 1, Sep. 2011.

[2] T. Li, S. Hazra, and W. Seah, "A position-based routing protocol for metropolitan bus networks," in *Vehicular Technology Conference, 2005. VTC 2005-Spring. 2005 IEEE 61st*, 2005.

[3] C. Lochert, M. Mauve, H. Fussler, and H. Hartenstein, "Geographic routing in city scenarios," *SIGMOBILE Mob. Comput. Commun. Rev.*, vol. 9, no. 1, pp. 69–72, Jan. 2005.

[4] J. Zhao and G. Cao, "Vadd: Vehicle-assisted data delivery in vehicular ad hoc networks," *Vehicular Technology, IEEE Transactions on*, vol. 57, no. 3, pp. 1910 –1922, may 2008.

[5] G. Samara, W. Al-Salihy, and R. Sures, "Security issues and challenges of vehicular ad hoc networks (vanet)," in *New Trends in Information Science and Service Science (NISS), 2010 4th International Conference on*, may 2010, pp. 393 –398.

[6] H. Saleet, O. Basir, R. Langar, and R. Boutaba, "Region-based location-service-management protocol for vanets," *Vehicular Technology, IEEE Transactions on*, vol. 59, no. 2, pp. 917–931, 2010.

[7] T. Leinmüller, C. Maihöfer, E. Schoch, and F. Kargl, "Improved security in geographic ad hoc routing through autonomous position verification," in *International workshop on Vehicular ad hoc networks - VANET*. ACM, 2006.

[8] T. Leinmuller, E. Schoch, and F. Kargl, "Position verification approaches for vehicular ad hoc networks," *Wireless Communications, IEEE*, vol. 13, no. 5, pp. 16–21, 2006.

[9] N. Bißmeyer, C. Stresing, and K. Bayarou, "Intrusion detection in vanets through verification of vehicle movement data," in *Vehicular Networking Conference (VNC), IEEE*, 2010.

[10] L. Dolberg, J. François, and T. Engel, "Efficient multidimensional aggregation for large scale monitoring," in *Proceedings of the 26th international conference on Large Installation System Administration: strategies, tools, and techniques*. USENIX Association, 2012.

[11] R. Burgess, "Google maps gets real-time traffic, crowdsources android gps data – accessed on march 5th 2013." [Online]. Available: http://www.techspot.com/news/48015-google-maps-gets-real-time-traffic-crowdsources-android-gps-data.html

[12] T. Leinmuller, R. Schmidt, E. Schoch, A. Held, and G. Schafer, "Modeling roadside attacker behavior in vanets," in *GLOBECOM Workshops, 2008 IEEE*, 2008.

[13] J. François, H. Abdelnur, R. State, and O. Festor, "Automated behavioral fingerprinting," in *International Symposium on Recent Advances in Intrusion Detection – RAID*. Springer-Verlag, 2009.

[14] J. Francois, H. Abdelnur, R. State, and O. Festor, "Ptf: Passive temporal fingerprinting," in *Integrated Network Management (IM), 2011 IFIP/IEEE International Symposium on*, 2011.

[15] S. Čapkun, L. Buttyán, and J.-P. Hubaux, "Sector: secure tracking of node encounters in multi-hop wireless networks," in *Workshop on Security of ad hoc and sensor networks*, ser. SASN '03. ACM, 2003.

[16] N. Sastry, U. Shankar, and D. Wagner, "Secure verification of location claims," in *Workshop on Wireless security - WiSe*. ACM, 2003.

[17] B. Xiao, B. Yu, and C. Gao, "Detection and localization of sybil nodes in vanets," in *Workshop on Dependability issues in wireless ad hoc networks and sensor networks – DIWANS*. ACM, 2006.

[18] C. Chen, X. Wang, W. Han, and B. Zang, "A robust detection of the sybil attack in urban vanets," in *International Conference on Distributed Computing Systems Workshops*. IEEE Computer Society, 2009.

[19] G. Yan, S. Olariu, and M. C. Weigle, "Providing vanet security through active position detection," *Comput. Commun.*, vol. 31, no. 12, 2008.

[20] J. Wang and N. Jiang, "A simple and efficient security scheme for vehicular ad hoc networks," in *Network Infrastructure and Digital Content, 2009. IC-NIDC 2009. IEEE International Conference on*, nov. 2009, pp. 591 –595.

[21] M. Raya, P. Papadimitratos, and J.-P. Hubaux, "Securing vehicular communications," *Wireless Communications, IEEE*, vol. 13, no. 5, pp. 8 –15, october 2006.

[22] H.-C. Hsiao, A. Studer, C. Chen, A. Perrig, F. Bai, B. Bellur, and A. Iyer, "Flooding-resilient broadcast authentication for vanets," in *Annual international conference on Mobile computing and networking*, ser. MobiCom. ACM, 2011.

[23] "Trial-Use standard for wireless access in vehicular environments - security services for applications and management messages," *IEEE Std 1609.2*, 2006.

[24] A. Studer, M. Luk, and A. Perrig, "Efficient mechanisms to provide convoy member and vehicle sequence authentication in vanets." in *SecureComm*. IEEE, 2007.

[25] T. H.-J. Kim, A. Studer, R. Dubey, X. Zhang, A. Perrig, F. Bai, B. Bellur, and A. Iyer, "Vanet alert endorsement using multi-source filters," in *International workshop on VehiculAr InterNETworking*, ser. VANET. ACM, 2010.

[26] C. Laurendeau and M. Barbeau, "Probabilistic localization and tracking of malicious insiders using hyperbolic position bounding in vehicular networks," *EURASIP J. Wirel. Commun. Netw.*, 2009.

[27] J.-H. Song, V. Wong, and V. C. M. Leung, "Secure location verification for vehicular ad-hoc networks," in *Global Telecommunications Conference - GLOBECOM IEEE*, 2008.

[28] P. Golle, D. Greene, and J. Staddon, "Detecting and correcting malicious data in vanets," in *International workshop on Vehicular ad hoc networks*, ser. VANET. ACM, 2004, pp. 29–37.

[29] P. Zhang, Z. Zhang, and A. Boukerche, "Cooperative location verification for vehicular ad-hoc networks," in *International Conference on Communications (ICC)*. IEEE, 2012, pp. 37–41.

[30] S. Ma, O. Wolfson, and J. Lin, "A survey on trust management for intelligent transportation system," in *International Workshop on Computational Transportation Science*, ser. CTS. ACM, 2011.

[31] N. Bißmeyer, J. Njeukam, J. Petit, and K. M. Bayarou, "Central misbehavior evaluation for vanets based on mobility data plausibility," in *International workshop on Vehicular inter-networking, systems, and applications*, ser. VANET. ACM, 2012.

[32] C. Wagner, J. Franois, R. State, and T. Engel, "Danak: Finding the odd!" in *nternational Conference on Network and System Security - NSS*. IEEE.

[33] L. Dolberg, J. François, and T. Engel, "Multi-dimensional aggregation for dns monitoring," in *Conference on Local Computer Networks – LCN*. IEEE, 2013.

[34] R. De La Briandais, "File searching using variable length keys," in *Papers presented at the the March 3-5, 1959, western joint computer conference*, ser. IRE-AIEE-ACM '59 (Western). New York, NY, USA: ACM, 1959, pp. 295–298. [Online]. Available: http://doi.acm.org/10.1145/1457838.1457895

[35] M. Behrisch, L. Bieker, J. Erdmann, and D. Krajzewicz, "Sumo-simulation of urban mobility-an overview," in *SIMUL 2011, The Third International Conference on Advances in System Simulation*, 2011, pp. 55–60.