CAN-Trace Attack: Exploit CAN Messages to Uncover Driving Trajectories

Xiaojie Lin, Baihe Ma, Xu Wang, *Member, IEEE*, Guangsheng Yu, Ying He, *Senior Member, IEEE*, Wei Ni, *Fellow, IEEE*, and Ren Ping Liu, *Senior Member, IEEE*

***Abstract*—Driving trajectory data remains vulnerable to privacy breaches despite existing mitigation measures. Traditional methods for detecting driving trajectories typically rely on map-matching the path using Global Positioning System (GPS) data, which is susceptible to GPS data outage. This paper introduces CAN-Trace, a novel privacy attack mechanism that leverages Controller Area Network (CAN) messages to uncover driving trajectories, posing a significant risk to drivers' long-term privacy. A new trajectory reconstruction algorithm is proposed to transform the CAN messages, specifically vehicle speed and accelerator pedal position, into weighted graphs accommodating various driving statuses. CAN-Trace identifies driving trajectories using graph-matching algorithms applied to the created graphs in comparison to road networks. We also design a new metric to evaluate matched candidates, which allows for potential data gaps and matching inaccuracies. Empirical validation under various real-world conditions, encompassing different vehicles and driving regions, demonstrates the efficacy of CAN-Trace: it achieves an attack success rate of up to 90.59% in the urban region, and 99.41% in the suburban region.**

***Index Terms*—Driving trajectory, Controller Area Network, road network, map-matching, subgraph matching.**

I. Introduction

Location privacy is a serious topic in the rapidly developing field of Intelligent Transportation Systems (ITS), attracting significant attention from both industry and academia [1], [2]. Sensitive location information, such as a driver's home or work address, is increasingly at risk due to privacy breaches in the ITS ecosystem. In the ITS ecosystem, vehicles can share location and motion data among themselves and with broader in-vehicle networks [3]–[5]. The misuse of driver location data can reveal driving trajectories and the geographical locations of drivers, thereby compromising the personal privacy of drivers [6], [7].

Map-matching process, which identifies a vehicle's location on a road network, is pivotal in these potential privacy breaches [8]–[10]. These attacks allow adversaries to accurately track and pinpoint vehicle locations using transmitted data [11]. Traditional methods to detect the driving trajectory have largely been built around Global Positioning System (GPS) [12] or a combination of GPS and additional sensors like the on-board camera [13], [14], Inertial Measurement Units (IMU) [15], [16] or On-Board Diagnostics (OBD) devices [17]. However, the traditional methods suffer from weaknesses such as data loss, GPS outages, and restricted access, thus offering loopholes for privacy attacks [18]–[20].

Advanced techniques have evolved to refine the map-matching process by incorporating mobile magnetometers [21]. The mobile magnetometers can provide granular vehicle tracking by identifying unique vehicle movements like stops, turns, and lane changes. However, the method using mobile phones faces the challenge of performance degradation when the position is shifted by the vehicle's occupants, whether intentionally or unintentionally. The variability in phone positioning further complicates the map-matching process and decreases the robustness of these methods against privacy attacks [21].

Controller Area Network (CAN) is a vehicle communication protocol, which defines the transmission of vehicle motion and control messages, such as vehicle speed, acceleration, and steering angle, allowing vehicular Electronic Control Units (ECUs) to communicate with each other within the in-vehicle network [22]–[24]. CAN messages offer a wealth of real-time and precise vehicle motion data, effectively mitigating the risks associated with data outage and integrity issues [25], [26]. CAN messages can be easily accessed through interfaces such as the OBD-II port [27]–[29], commonly exploited by third-party car diagnostic software, OBD-II applications, and OEM assistance applications.

In this paper, we pioneer a CAN-centric driving trajectory tracking strategy that serves as an alternative to the map-matching methods reliant on GPS or magnetometers. We use basic CAN messages as an innovative data source to decipher driving trajectories in the form of weighted line graphs. Subsequently, we reframe the problem of trajectory identification as the task of pinpointing the corresponding subgraph, derived from CAN messages, in a targeted road network. The key contributions are as follows:

- We propose a novel trajectory reconstruction algorithm that composes weighted graphs to depict driving trajectories. The algorithm discerns key vehicle movements, i.e., stops, turns, and driving lengths, using only speed and pedal CAN messages.
- We are the first to introduce the graph-based matching technique for driving trajectory identification from road networks. We also design a new metric to evaluate matching results, accounting discrepancies between the CAN perspective and the actual road network.
- We validate the proposed CAN-Trace attack via extensive real-world experiments across diverse road networks and

X. Lin, B. Ma, X. Wang, Y. He, and R. P. Liu are with the Global Big Data Technologies Centre, University of Technology Sydney, Australia. E-mail: {xiaojie.lin, baihe.ma, xu.wang, ying.he, renping.liu}@uts.edu.au

G. Yu and W. Ni are with Data61 CSIRO, Sydney, Australia. E-mail: {saber.yu, wei.ni}@data61.csiro.au

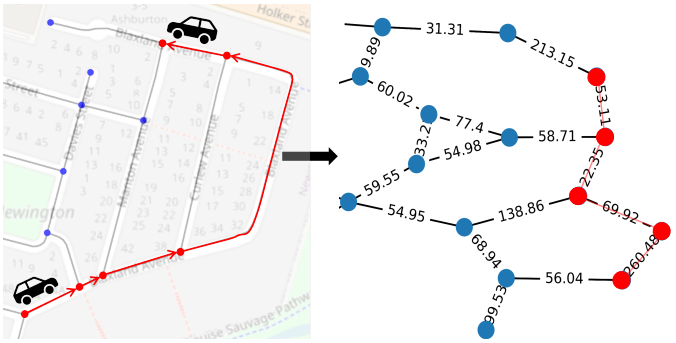


Fig. 1. Road network to graph: a road network is converted from the 2D street map into a graph $G = (V, E)$, where edges represent road segments and the weight of an edge indicates the length of the corresponding road segment. The actual driving trajectory can be represented as a subgraph $G^* = (V^*, E^*)$. Blue nodes indicate road intersections not covered by the driving trajectory, while red nodes denote intersections in the driving trajectory.

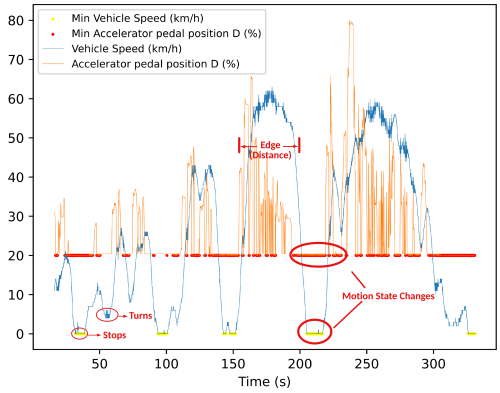


Fig. 2. CAN to driving trajectory: vehicle stops or turns can be identified as nodes by analyzing the vehicle speed and pedal position data in OBD response, and the length of the road segments can be calculated by the distance between two adjacent nodes.

vehicles, demonstrating success rates up to 90.59% in the urban region and 99.41% in the suburban region.

The rest of the paper is organized as follows: Section II gives the overview of the system and attack models and Section III describes each process of the CAN-Trace attack in detail. Section IV presents the evaluation of the CAN-Trace attack in the real-world environment. Section V summarizes the related work. Section VI concludes this paper.

II. System and Attack Models

The proposed CAN-Trace attack considers road networks and deduces the driving trajectory of the on-road vehicle by collecting and analyzing CAN messages through a driving period. Table I summarizes the notation used throughout this paper. The system model is described as follows:

- **Attack Target**: The attack target is to expose the real-time geolocation of the driver and identify the driving trajectory.
- **Adversary Capability**: Adversaries have full prior knowledge of the road networks, i.e., where the target vehicle is located within a city or a suburb size. Adversaries can collect vehicular motion data by analyzing the CAN message to infer vehicle activities on road networks, e.g., stops or turns.

TABLE I
Notation and Definitions

Notation	Description
G	Data graph of the road network
G^*	Graph of the actual driving trajectory
$\bar{R}$	Graph of the constructed driving trajectory
$\hat{R} = \{r_k\}$	Set of matched subgraphs r_k
E / E^*/ E^R / E^{r_k}	Set of edges in G / G^* / $\bar{R}$ /r_k
V / V^*/ V^R / V^{r_k}	Set of nodes in G / G^* / $\bar{R}$ /r_k
$w_{i,j}$ / $w^*_{i,j}$ / $w^R_{i,j}$ / $w^k_{i,j}$	Weight of $e_{i,j}$ / $e^*_{i,j}$ / $e^R_{i,j}$ / $e^{r_k}_{i,j}$
$\mathcal{M} = \{M^s, M^p\}$	Set of CAN messages of OBD response data
$M^s = \{m^s_i, t^s_i\}$	Set of OBD response of vehicle speed value m^s_i and the timestamp t^s_i
$M^p = \{m^p_i, t^p_j\}$	Set of OBD response of accelerator pedal position value m^p_i and the timestamp t^p_j
N^s_T / N^p_T	Number of collected OBD messages of vehicle speed/pedal position
N'_k	Number of correctly inferred nodes in r_k
Q^*	Number of nodes in the graph of actual driving trajectory
Δ^s / Δ^p	Time difference threshold to determine the nodes of vehicle stops/turns
σ	Relative tolerance of edge weight used in the matching process
θ	Distance difference of the $e^R_{i,j}$ and $e^k_{i,j}$ pair of $\bar{R}$ and r_k
P	Attack precision
Ψ	Attack success rate
D	Spatial distance offset between the deduced and the actual driving trajectory
$\mathbb{F}$	False negative rate

A. Prior Knowledge

1) Road Network: A road network can be extracted from a Two-Dimensional (2D) street map and converted into an undirected weighted graph $G = (V, E)$, which contains the geospatial information of streets. The set of nodes, V, represents the road intersections in the road network, while the set of edges, E, are the segments that connect adjacent nodes of V within the road network. The weights of edges are the length of the segments in the road network.

As shown in Fig. 1, the actual driving trajectory is regarded as a graph $G^* = (V^*, E^*)$ that is a subgraph of G. The road intersections passed by the car are denoted as V^*, and the road segments passed are denoted as E^*. The problem of detecting the actual driving trajectory is then converted into that of finding the corresponding subgraph G^* within G by matching the graph constructed from CAN messages.

2) CAN Messages: The adversaries obtain OBD response $\mathcal{M} = \{M^s, M^p\}$ to construct the line graph of the driving path $\bar{R}$ as $\mathcal{M} \rightarrow \bar{R} = (V^R, E^R)$. Analyzing the motion data embedded in $\mathcal{M}$, the adversaries identify the vehicle stops or turns as the nodes V^R, and the path between adjacent nodes as edge E^R. CAN-Trace attack leverages the CAN messages of vehicle speed $M^s = \{(m^s_i, t^s_i)\}$ (where $i \leq N^s_T$) and those

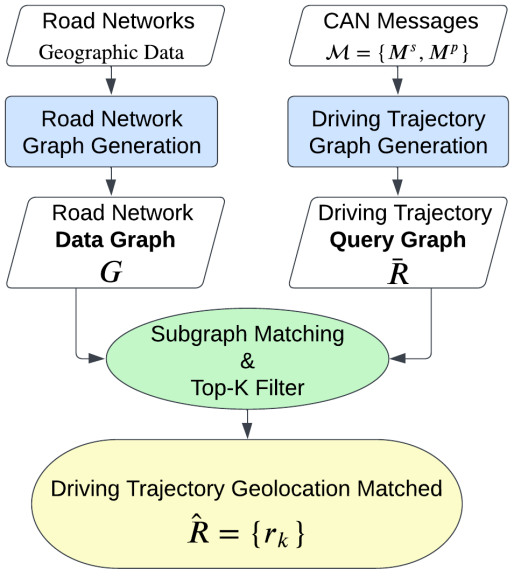


Fig. 3. Attack model: the CAN-Trace attack identifies the driving trajectory by matching the driving path graph $\bar{R}$ with the road network graph G. The road network graph generation converts geographic data of road networks into G using OpenStreetMap. The driving trajectory graph generation leverages the CAN messages to transform vehicle motion data into $\bar{R}$. The generated $\bar{R}$ is further matched with G using a subgraph matching algorithm to find subgraphs r_k, i.e., driving trajectory candidates. Top-K ranking rule is utilized to select a specified number of deduced subgraph candidates as the attack result $\hat{R}$.

of the accelerator pedal position $M^p = \{(m_j^p, t_j^p)\}$ (where $j \leq N_T^p$) to construct the line graph of the driving path. Here, (m_i^s, t_i^s) is the i-th vehicle speed message of the vehicle speed value m_i^s at the time t_i^s, and (m_j^p, t_j^p) is the j-th vehicle pedal position message of the pedal position value m_i^p at the time t_j^p. N_T^s and N_T^p are the number of messages in M^s and M^p, respectively. As demonstrated in Fig. 2, vehicle activities such as stops and turns can be identified from the OBD response statistics.

III. CAN-Trace Attack

To the best of our knowledge, the CAN-Trace attack is the first to propose the OBD-II port as a new attack interface and the CAN messages as the data source. As demonstrated in Fig. 3, the CAN-Trace attack consists of the driving trajectory graph generation, the road network graph generation, the subgraph matching, and the Top-K ranking. The CAN-Trace attack constructs the driving trajectory graph $\bar{R}$ based on vehicle stops and turns from CAN messages. Then, it applies subgraph matching algorithms to match $\bar{R}$ with the road network graph G. Since more than one subgraph candidate r_k can be matched by the CAN-Trace attack, ranking rules are applied to identify the Top-K subgraphs. Algorithms 1 and 2 illustrate the attack method.

A. Graph Generation

To the best of our knowledge, we are the first to generate a road network graph by using the CAN messages to infer the driving trajectory. The detailed steps are discussed as follows.

*1) **Road Network Data Graph**:* Vehicles are subject to traffic rules and are constrained to drivable road segments in the road network $G = (V, E)$. Nodes V and edges E are constructed to generate a purified G as given by:

- **Nodes**: The i-th node v_i stands for the i-th entry or exit of a road segment in the road network. A node can have multiple entries or exits, e.g., a roundabout or intersection.
- **Edges**: An edge $e_{i,j}$ exists if there is a route between v_i and v_j. The value of $e_{i,j}$ is the route length of the road segment between v_i and v_j.

The actual driving trajectory is located within the road network, which means G^* is the subgraph of G satisfying $V^* \subseteq V$ and $E^* \subseteq E$.

*2) **Graph of Constructed Driving Trajectory**:* Adversaries generate the graph of the constructed driving trajectory $\bar{R} = (V^R, E^R)$ using the OBD response set $\mathcal{M} = \{M^s, M^p\}$ of a driving period T. During the period, N_T^s speed messages and N_T^p pedal position messages are collected, so we have $i \leq N_T^s$ and $j \leq N_T^p$ for any m_i^s and m_j^p in $\mathcal{M}$.

Nodes V^R refer to the vehicle stops and turns embedded in the OBD response $\mathcal{M} = \{M^s, M^p\}$. Vehicle status switches between driving and stopping or turning are determined as follows:

- **Stops at traffic signals or signs**: $m_i^s = 0$.
- **Turns at the road intersections**: $m_j^p = \min(m^p)$.

The minimum value of the pedal position indicates the driver releases the throttle. Note that the default value of the pedal position is not equal to zero. We assume that the vehicle turns at the road intersection when the minimum pedal position lasts for a period.

As illustrated in Fig. 4(a), nodes V^R can be identified from CAN messages by analyzing the data points and factors like the time difference. Details of the node selection in constructing the graph are found in Lines 1–5 of Algorithm 1. The CAN-Trace attack feeds the set of all $m^{s=0}$ as P^s and $m^{p=\min(p)}$ as P^p. Leveraging the clustering algorithm like K-Means, CAN-Trace attack clusters the time difference between two consecutive candidates in P^s and P^p to find the stopping and turning clusters. The time differences are calculated by $d(P_i^s, P_{i+1}^s) = t_{i+1}^s - t_i^s$ for P^s and $d(P_j^p, P_{j+1}^p) = t_{j+1}^p - t_j^p$ for P^p, respectively. The clusters distinguishing nodes and non-nodes usually have a long distance between two clusters, where the CAN-Trace attack selects the upper bound of the small cluster as the delimit. The delimits of the identified cluster of stops and turns are used as the threshold Δ^s and Δ^p to exclude the redundant nodes for stops and turns, respectively. Δ^s is introduced to eliminate the messages of the short-term start and stop, and Δ^p is used to avoid the messages of the lane changes.

The method to construct the nodes of the line graph of the driving trajectory $\bar{R}$ is described in Algorithm 1 from Lines 6–26. The node of a vehicle stop can be identified when the time difference between two data points in M^s is greater than or equal to Δ^s and the two points satisfy $m_i^s = 0$ and $m_j^s = 0$. The node of a vehicle turning can be determined when the time

Algorithm 1: CAN to Graph Conversion

Input : CAN messages $\mathcal{M} = \{M^s, M^p\}$
Output: Graph of constructed driving trajectory $\bar{R} = (V^R, E^R)$
▷ **Initialization**
/* Put data points whose $m^s = 0$ or $m^p = \min(m^p)$ into the candidate sets P^s and P^p. */
$P^s \leftarrow \{m^{s=0}\}$, $P^p \leftarrow \{m^{p=\min(m^p)}\}$;
/* Use the time gap between two consecutive candidates for clustering. */
$T^s = \{d(P^s_i, P^s_{i+1})\}$, $T^p = \{d(P^p_j, P^p_{j+1})\}$;
/* Cluster T^s and T^p for thresholds Δ^s and Δ^p. */
Cluster(T^s), **Cluster**(T^p);
$\Delta^s \leftarrow$ **UpperBoundOfStoppingClusters**(T^s);
$\Delta^p \leftarrow$ **UpperBoundOfTurningClusters**(T^p);
▷ **Graph Construction**
/* Initial index i and j for the loop on M^s. */
$i, j = 1$;
for $j < N^s_T$ **do**
  if $m^s_j = 0$ **then**
    if $t^s_j - t^s_i \geq \Delta^s$ **then**
      V^R.**append**$((m^s_j, t^s_j))$;
    end
    $i = j$
  end
  $j = j + 1$
end
/* Initial index i for the loop to parse M^p. */
$i, j = 1$;
for $j < N^p_T$ **do**
  if $m^p_j = \min(m^p)$ **then**
    if $t^p_j - t^p_i \geq \Delta^p$ **then**
      V^R.**append**$((m^p_j, t^p_j))$;
    end
    $i = j$;
  end
  $j = j + 1$;
end
/* Sort V^R based on the message timestamp. */
V^R.**sort**();
/* Merge the duplicated nodes. */
for v^R_k *and* v^R_{k+1} *in* V^R **do**
  Calculate the weight of edge $w^R_{k,k+1}$; ▷ with (1)
  /* Select the shortest road network segment in G denoted as $\min(w_{i,j})$ */
  if $w^R_{k,k+1} < \min(w_{i,j})$ **then**
    /* Remove nodes whose edges less than $\min(w_{i,j})$. */
    V^R.delete(v^R_k);
  end
end
/* Construct $\bar{R}$ with the merged V^R */
$E^R \leftarrow \{e^R_{k,k+1}\}$ where $v_k, v_{k+1} \in V^R$;
return $\bar{R} = (V^R, E^R)$

Algorithm 2: Driving Trajectory Detection: Matching and Filtering

Input : Road network graph G, graph of constructed driving trajectory $\bar{R}$, relative tolerance σ, Top-K value K
Output: A set of detected driving trajectory $\hat{R} = \{r_k\}$
/* Set different tolerance σ in matching process to find l subgraphs until $l \geq K$ with $\max(\sigma)$ */
Set σ and K in matching $\bar{R}$ with G to identify subgraph candidate r_l;
Calculate the difference of edge pair $\varpi^l_{i,j} = |w^l_{i,j} - w_{i,j}|$;
if $\forall\ \varpi^l_{i,j} \leq \sigma \times w_{i,j}$ **then**
  /* Append matched nodes and edges to r_l */
  $V^{r_l} \leftarrow v_i, v_j$;
  $E^{r_l} \leftarrow e^{r_l}_{i,j}$;
  Calculate the weight of $e^{r_l}_{i,j}$ as $w^l_{i,j}$; ▷ with (1)
end
/* Calculate the average distance difference between the edge pair of line graph $\bar{R}$ and r_l as θ_l */
$\theta_l \leftarrow \sum_{i=1}^{Q^*-1} \frac{|w^R_{i,i+1} - w^l_{i,i+1}|}{Q^*-1}$; ▷ with (3)
/* Sort r_l by θ_l as $\{r_k\}$ */
$\{r_k\} \leftarrow$ **Sort**(r_l, θ_l);
/* Select K subgraphs as $\hat{R}$ */
return $\hat{R} = \{r_k\}, k \leq K$

difference between two data points in M^p is greater than or equal to Δ^p, and the two points satisfy $m^p_i = \min(m^p)$ and $m^p_j = \min(m^p)$. The nodes in V^R are filtered out and built. Then, all the nodes will be sorted by the message timestamp t^R.

The redundant nodes in V^R need to be merged before constructing the edges E^R since the same node can be generated twice from the vehicle speed and pedal position separately. The duplicated nodes in V^R have the following situations:

- **Short-term start and stop before red lights:** fully stop and short-term start when the driver quickly pushes and releases the throttle can lead to the repeated nodes.
- **Pedal releasing before stop signs:** the driver may release the throttle before the vehicle fully stops, i.e., $t^{\min(p)}_j < t^{s=0}_i$ of the duplicated nodes v^R_i and v^R_j.

As given by Lines 27–32 in Algorithm 1, all duplicated nodes are merged by checking the edge weight between two adjacent nodes v^R_k and v^R_{k+1} in V^R. The edge weight $w^R_{k,k+1}$ stands for the identified driving path connected by the two consecutive nodes, i.e., the road segment between v^R_k and v^R_{k+1}. As the driving trajectory is located in a certain road network area, any edge weight should be no less than the length of the shortest road segments in the road network area, i.e., $\forall w^R_{k,k+1} \geq \min(w_{i,j})$. Once the edge weight of $w^R_{k,k+1}$ does not satisfy the condition, the CAN-Trace attack will merge the node by deleting the node v^R_k.

Edges E^R, representing the driving path between two nodes, are constructed by connecting the consecutive nodes in V^R after merging nodes. Since the constructed graph of the driving trajectory from CAN messages is a line graph, each node is connected with only one or two nodes in the graph. The weight $w^R_{i,i+1}$ of the edge $e^R_{i,i+1}$ is the road length between nodes v_i and v_{i+1}, where v_i and v_{i+1} are the i-th and $(i+1)$-th intersections, respectively. CAN-Trace attack calculates the distance traveled by the vehicle using discrete data points of vehicle speed and corresponding timestamps. The weight $w^R_{i,i+1}$ can be calculated by using the rectangle method to estimate the driving distance, which is as given by

$$w^R_{i,i+1} = \sum_{t_a \leq t_l < t_b} m^s_l \times (t_{l+1} - t_l), \tag{1}$$

where t_a and t_b are the timestamps when the vehicle leaves the node v_i and arrives the node v_{i+1}, respectively. The timestamp

of the CAN messages between v_i and v_{i+1} when the vehicle is driving during the period from t_a to t_b is denoted as t_l. The vehicle speed at a given timestamp t_l is denoted as m_l^s. The time difference between two consecutive timestamps t_l and t_{l+1} is represented as $(t_{l+1} - t_l)$. The driving length between two consecutive nodes v_i and v_{i+1}, i.e., $w_{i,i+1}^R$, is calculated by multiplying the vehicle speed m_l^s with the time difference $(t_{l+1} - t_l)$.

The driving trajectory construction from CAN messages is detailed in Algorithm 1 and illustrated in Fig. 4(b). The CAN-Trace attack reconstructs the driving trajectory by leveraging vehicle motion data from CAN messages, specifically vehicle speed and pedal position. CAN-Trace begins by determining speed and pedal thresholds, i.e., Δ^s and Δ^p, using the K-means clustering algorithm on the respective CAN messages. Next, CAN-Trace identifies node candidates by selecting zero-speed CAN messages and minimal-pedal CAN messages when the time differences exceed the identified thresholds, as shown in Fig. 4(b). The zero-speed and minimal-pedal CAN messages are verified, as they imply that the vehicle is either stopping at or crossing intersections. The distance between node candidates is calculated using the speed data and serves as the edge weight between them. CAN-Trace then removes edges whose weights are shorter than the shortest road segment in the road network, merging the end nodes of these edges accordingly.

B. Proposed Trajectory Matching and Detection

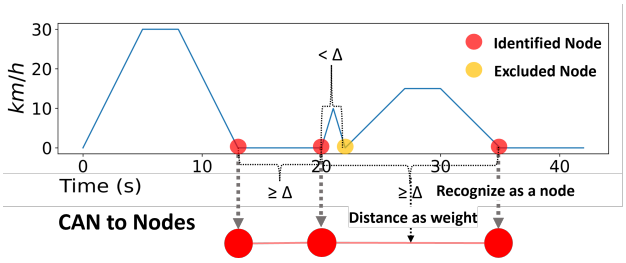
By using the generated graph, the CAN-Trace attack employs a subgraph matching algorithm to find the isomorphic subgraph of G. The matching process can be formulated as given by

$$\hat{\mathcal{R}} = F(G, \bar{R}) = \{r_k\}, \tag{2}$$

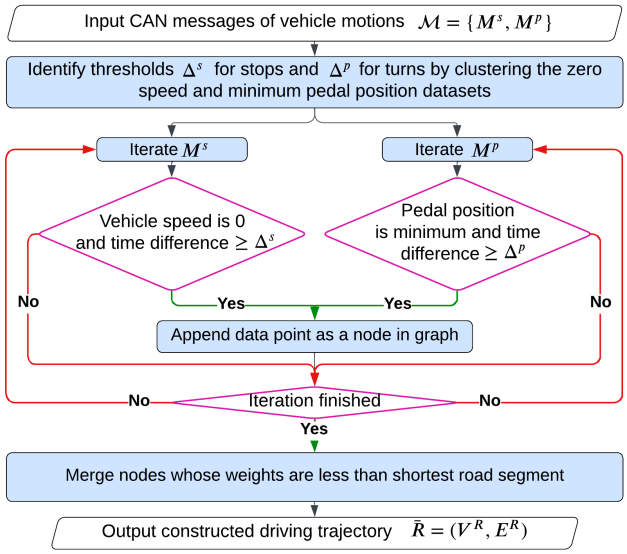
where $\hat{\mathcal{R}}$ is the set of the matched subgraphs. r_k is the k-th subgraph in $\hat{\mathcal{R}}$ and F is the matching function, e.g., Vento-Foggia 2 (VF2) [30]. VF2 is an improved matching method that employs a depth-first search strategy and utilizes feasibility rules to prune the search space when finding subgraph isomorphism. With less memory requirements, VF2 is capable of effectively matching graphs of thousands of nodes and edges. In the proposed attack, the VF2 is employed due to its efficiency. By matching the road network graph G with the constructed graph from CAN messages $\bar{R}$, the matched subgraphs r_k can infer the driving trajectory with the geolocation data.

As given by Lines 1–7 in Algorithm 2, the subgraph $\hat{R}$ is identified by matching and sorting the subgraph candidates r_l. Note that the lengths of road segments in the network graph G and the constructed graph $\bar{R}$ can differ due to various factors, such as inaccurate map data and different driving behaviors.

Relative Tolerance. The relative tolerance σ bounds the tolerated value of the edge weight when matching the $\bar{R}$ with G to identify the subgraph candidates r_l. The tolerated length is calculated as $\sigma \times w_{i,i+1}$. The weight difference between the edge pair of $e_{i,j}^{r_l}$ and $e_{i,j}$ is examined as $\varpi_{i,j}^l$. The CAN-Trace attack finds the subgraph candidates r_l when $\forall\ \varpi_{i,i+1}^l \leq \sigma \times w_{i,i+1}$.



(a) Trajectory graph construction from vehicle speed to stops



(b) Algorithm flowchart of graph construction from CAN messages

Fig. 4. Graph construction: nodes are parsed and extracted from CAN messages. CAN-Trace attack calculates the time differences between two data points that are possible node candidates. Nodes are identified when the time differences are greater than the examined thresholds. The distance between two nodes is calculated as the edge weight. The x-axis in (a) represents the elapsed driving time in seconds, while the y-axis indicates the vehicle speed in kilometers per hour. The algorithm flowchart in (b) represents the workflow of constructing graphs by parsing CAN messages and identifying nodes of stops and turns.

Top-K ranking. The Top-K ranking method is used to sort and filter the identified subgraphs r_l into the set $\hat{R} = \{r_k\}$, as referencing to Lines 8–10 in Algorithm 2. The CAN-Trace attack calculates the distance difference between the edge pair of the driving path of $\bar{R}$ and the matched subgraph of r_l, i.e., $w_{i,i+1}^R - w_{i,i+1}^l$. The mean of the differences for the entire driving trajectory, denoted as θ_l, is assessed to rank the matched subgraphs, as given by

$$\theta_l = \sum_{i=1}^{Q^*-1} \frac{|w_{i,i+1}^R - w_{i,i+1}^l|}{Q^* - 1}, \tag{3}$$

where $w_{i,i+1}^R$ is the weight of the driving path $e_{i,i+1}^R$, and $w_{i,i+1}^l$ is the weight of the matched path $e_{i,i+1}^{r_l}$. The number of nodes in the actual driving trajectory graph is denoted as Q^*. With the setting of K, CAN-Trace attack ranks r_l by θ_l to filter out K subgraphs r_k as the attack results $\hat{R} = \{r_k\}$, where $k \leq K$.

C. Complexity Analysis

The proposed attack includes the sequential execution of CAN to graph conversion (i.e., Algorithm 1) and trajectory detection (i.e., Algorithm 2). In Algorithm 1 Line 3, the K-Means clustering step for finding the thresholds Δ^s and Δ^p operates at $O(N_T^s \log N_T^s + N_T^p \log N_T^p)$ [31]. The complexity of the graph construction steps in Algorithm 1 Lines 6–15, Lines 16–25, and Lines 27–32, are $O(N_T^s)$, $O(N_T^p)$, and $O(V^R)$, respectively, where N_T^s and N_T^p are the numbers of collected CAN messages of vehicle speed and pedal, respectively. The sorting step of Algorithm 1 Line 26 has the complexity of $O(V^R \log V^R)$, where V^R is the number of nodes in the constructed driving trajectory graph $\bar{R}$. Thus, the computational and time complexity of Algorithm 1 is $O((\log N_T^s + 1)N_T^s + (\log N_T^p + 1)N_T^p + V^R + V^R \log V^R)$. In practice, the vehicle speed and pedal position messages can be collected at the same rate, and the number of the vehicle messages is much greater than the number of stops in the driving trajectory, i.e., $N_T^s = N_T^p \gg V^R$. Thus, the complexity of Algorithm 1 can be written as $O(N_T^s \log N_T^s)$.

The most time-consuming step of Algorithm 2 is the subgraph matching. The VF2-based implementation exhibits a complexity of $O(N_V^2)$ in the best case and $O(N_V! N_V)$ in the worst case [30], where N_V is the number of nodes in the graph of the road network. As a result, the overall computational and time complexity of the proposed attack is between $O(N_T^s \log N_T^s + N_V^2)$ and $O(N_T^s \log N_T^s + N_V! N_V)$. We note that the proposed CAN-Trace attack can be conducted offline, with CAN messages collected during driving and processed afterward. Therefore, the attack is not time-critical. The above-mentioned complexity is tolerable.

D. Assessment Criteria

The node in r_k is determined as true positive when the geolocation is the same as the corresponding node in G^*. The attack performance is assessed by the attack success rate Ψ, attack precision P, and the spatial distance offset D.

The attack success rate Ψ refers to the attack coverage of the CAN-Trace attack, evaluated by

$$\Psi = \frac{\left|\bigcup_{i=1}^{k}(V^{r_k} \cap V^*)\right|}{Q^*}, \tag{4}$$

where V^{r_k} is the inferred nodes of a subgraph r_k, and V^* is the number of attempted attacks. The size of the distinct set containing all correctly inferred nodes in r_k is calculated as $\left|\bigcup_{i=1}^{k}(V^{r_k} \cap V^*)\right|$. The number of nodes in the graph of the actual driving trajectory G^* is denoted as Q^*.

The attack precision P refers to the positive predictive value, i.e., the True Positive (TP) divided by the sum of the true positive and False Positive (FP), which indicates the effectiveness of the attack. In our experiments, P represents the average percentage of the correctly matched nodes of the attack result and is as given by

$$P = \frac{TP}{TP + FP} = \frac{1}{K}\sum_{i=1}^{K}\frac{N_k'}{Q^*}, \tag{5}$$

where K is the number of matched subgraphs and N_i' is the number of correctly inferred nodes in r_i.

The spatial distance offset D quantifies the geospatial discrepancy between the inferred driving trajectory r_k and the actual driving trajectory G^*, which is given in (6).

$$D = \frac{1}{K}\sum_{k=1}^{K}\frac{d(V^*, V^{r_k})}{Q^*}, \tag{6}$$

where V^* is the set of nodes in the actual driving trajectory, and the V^{r_k} is the set of nodes in the selected subgraph r_k. $d(V^*, V^{r_k})$ is the average of the Euclidean distance between each node pair of the selected subgraph and that of the actual driving trajectory.

The false negative rate $\mathbb{F}$ is the average ratio of nodes in the actual driving trajectory G^* that are not correctly identified in the matched subgraph r_k, as given by

$$\mathbb{F} = \frac{1}{K}\sum_{i=1}^{K}\frac{N_k^{\mathbb{F}}}{Q^*}, \tag{7}$$

where $N_k^{\mathbb{F}}$ represents the number of nodes that are in the actual driving trajectory graph G^* but not identified in the matched subgraph r_k, and Q^* is the number of nodes in the graph of the actual driving trajectory.

IV. Performance Evaluation

This section evaluates the proposed CAN-Trace attack in a real road network under different experimental settings and subjects, including the type and area size of the road network, the models and years of experimental cars, and the length of the actual driving trajectory Q^*.

A. Experiment Setup

We collect driving data from three different cars (i.e., a 2013 petrol car, a 2015 petrol car, and a 2022 hybrid car). All three cars can provide the required vehicle motion data, i.e., speed and pedal data, via the standard OBD-II protocol. A laptop is connected to each vehicle's OBD-II port using the same CAN analyzer (specifically, the PEAK PCAN-USB Pro FD adapter[1]). The CAN analyzer reads vehicle speed and pedal data at 0.1-second intervals and logs the motion data on the laptop with the complementary software (i.e., PCAN-Explorer[2]). The collected CAN messages are converted from the raw trace format (.trc) into CSV format, with decoded numerical vehicle speed and pedal data, along with the timestamp of each data entry. The consistent data collection process and settings help minimize discrepancies attributable to different vehicles. While the current data collection process requires physical devices to be connected to the CAN bus, typically through the OBD-II port, car manufacturers and vehicle applications may have direct access to CAN messages.

[1]The PEAK PCAN-USB Pro FD adapter allows a computer to connect the vehicle CAN bus via the OBD-II port. https://www.peak-system.com/PCAN-USB-Pro-FD.366.0.html?&L=1

[2]PCAN-Explorer is a professional Windows software that equips with PEAK devices to view, transmit, and record real-time CAN messages. https://www.peak-system.com/PCAN-Explorer-6.415.0.html?&L=1

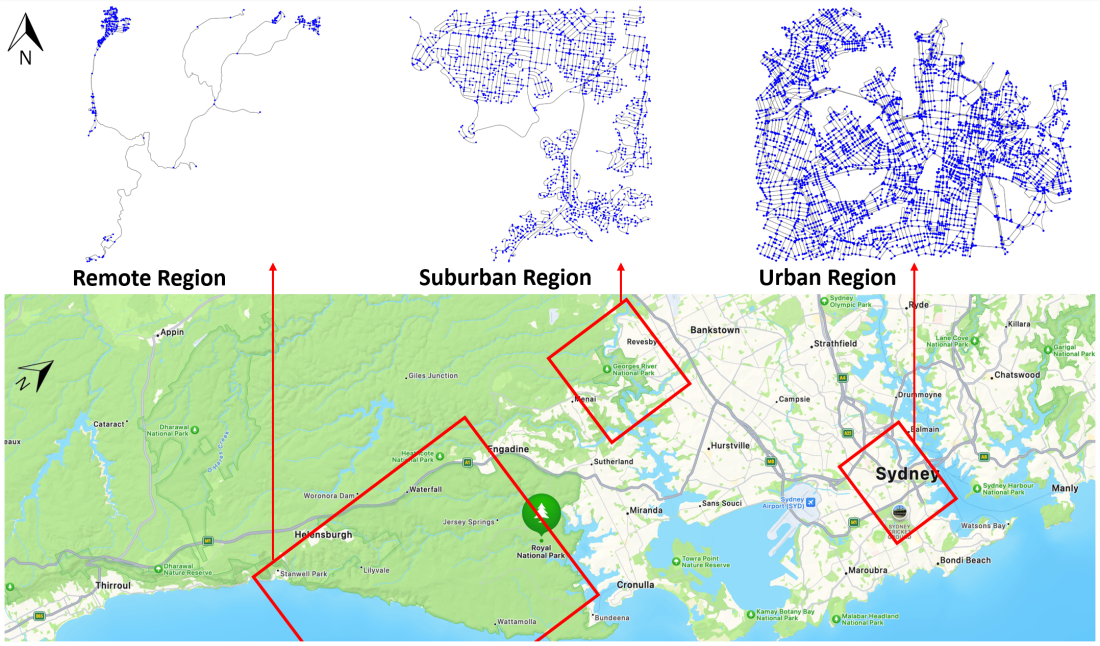


Fig. 5. Topology graphs of used road networks in remote, suburban, and urban regions, showing road network density from low to high. The road networks are from Sydney city, extracted using OpenStreetMap. Blue dots represent road network intersections, and edges represent the road segments connecting the intersections.

The proposed CAN-Trace method are tested across 279 driving trajectories in three types of regions within Sydney: remote, suburban, and urban, as illustrated in Fig. 5. The remote regions feature low density, long road segments, and distinct street blocks. The suburban regions have medium density, a mix of short and long road segments, and partially similar street blocks. The urban regions are characterized by high density, short road segments, and similar street blocks.

The matching process of the CAN-Trace attack is performed on a Windows 10 laptop with Intel(R) Core(TM) i7-1185G7 CPU@3GHz and 16G RAM. The server runs in an Anaconda environment with Python 3.6.8, NetworkX 2.5.1, OSMnx 1.2.1 and Numpy 1.19.5.

B. Experimental Setting

The experimental vehicle is driven in different types of road networks, i.e., urban and suburban, to evaluate the attack generalizability. The evaluation is also performed across diverse configurations, i.e., the road network area, the value of K, and the number of nodes in the driving trajectory graph Q^*.

Road Network Area. The road network area represents the size of the road network where the constructed trajectories from CAN messages are matched. A large road network area stands for a bigger graph and more nodes and edges. In this experiment, the road network area is set to various values, i.e., 5 km × 5 km, 10 km × 10 km, 15 km × 15 km, 20 km×20 km, 25 km×25 km, and 30 km×30 km. A 5 km×5 km road network area crosses two or more suburbs of the urban region and is only located in one suburb of the suburban region. The majority of the metro city of Sydney can be covered in a 30 km × 30 km road network area, which means our experimental settings can prove the attack performance regarding the victim's daily range of driving.

Driving Trajectory Length. The length of an actual driving trajectory can be captured by the number of nodes in the related graph, denoted by Q^*. The experiments have different settings of Q^* value: 5, 7, 10, 12 and 15. In our experiments, the driving trajectory length is approximately 100 m long for the edge between two nodes. Similar road network patterns are intensive in the urban region but sparse in the suburban region.

Value of K. Top-K ranking, which determines the size of $\hat{R}$, is designed to select K detected subgraph candidates $\hat{R} = \{r_k\}$ as the attack results. The attack results in $\hat{R}$ occasionally contain outliers (i.e., a subgraph candidate with zero nodes matched correctly) at smaller K values within a large road network area, which leads to a lower attack success rate Ψ. A higher K value considers a broader range of matched candidates to enhance attack robustness, but the trade-off is a lower attack precision P.

C. Experimental Results

The overall trend of all the attack success rate Ψ, attack precision P, and the spatial distance offset D are consistent in the urban and suburban region, as shown in Figs. 6–9.

*1) **Attack Success Rate**:* The impact of different K values is indicated in Fig. 6 where we compare Ψ of different K and discuss the related outcome under different road network areas or the driving trajectory length Q^*. The y-axes in Figs. 6(a) and 6(b) are the average values of Ψ among different driving trajectory lengths Q^*, while the y-axes in Figs. 6(c) and 6(d) are the average values of Ψ among different road network

areas. In urban and suburban road networks, Ψ progressively converges as the value of K increases and stabilizes for $K \geq 5$. A greater value of K comes with a bigger tolerance σ to accept the relative error of the edge weight. Therefore, the correct nodes and edges can be found by enlarging the relative error in the matching process. As shown in Figs. 6(a) and 6(b), Ψ can reach 90.59 % in urban and 99.41 % in suburban at $K = 10$ within the road network area of 5 km × 5 km. The result indicates that the CAN-Trace attack can deduce the majority location of a target driving trajectory within the top 10 subgraph candidates, showing a good attack efficiency. The CAN-Trace attack has a better attack success rate in the urban region. As shown in Figs. 6(a) and 6(b), the average values of Ψ are significantly higher in urban road networks, particularly in those with the road network area of 20 km×20 km, 25 km×25 km and 30 km×30 km. The upward trend is obvious when $K = 1$ with a 30 km × 30 km road network area, where Ψ increases from 36.88 % in suburban to 50.35 % in urban road networks.

The attack success rate Ψ shows the positive correlation with the Q^*, as indicated in Figs. 6(c) and 6(d). As shown in Figs. 6(c) and 6(d), the proposed attack can apply a small K to achieve good performance on Ψ with a longer driving trajectory length. Notably, the CAN-Trace attack can gain a higher attack success rate Ψ when $Q^* \geq 10$ in our experiments. The success rate Ψ in the urban region appears to be generally superior to that in the suburban region. This could potentially be attributed to the more distinct features of road network patterns in the urban region. In either urban or suburban regions, Ψ rises to reach around 90 % with a greater Q^*.

We proceed to compare the attack success rate of the proposed CAN-Trace method with the state-of-the-art trajectory detection methods[3], i.e., [21] and [32]. In comparison with [21], the proposed CAN-Trace achieves the attack success rates of 79.76% and 50.35% for 10 km × 10 km and 30 km×30 km areas, respectively, when $K = 1$. By contrast, the corresponding success rates in [21] for 100 km^2 and 900 km^2 areas are 61.1% and 33.5%, respectively. Compared to [32], CAN-Trace achieves similar success rates on 400 km^2 areas. Specifically, CAN-Trace reaches a 78.18% attack success rate for 20 km × 20 km areas when $K = 10$, as shown in Fig. 6(a). This success rate corresponds to a 97.73% partial match rate (which is obtained from $\frac{78.18\%}{80\%}$ based on the 80% matching requirement for partial match success rate in [32]), which is similar to the partial match success rate for the 400 km² area (i.e., Q1 results given in Fig. 8(b) of [32]).

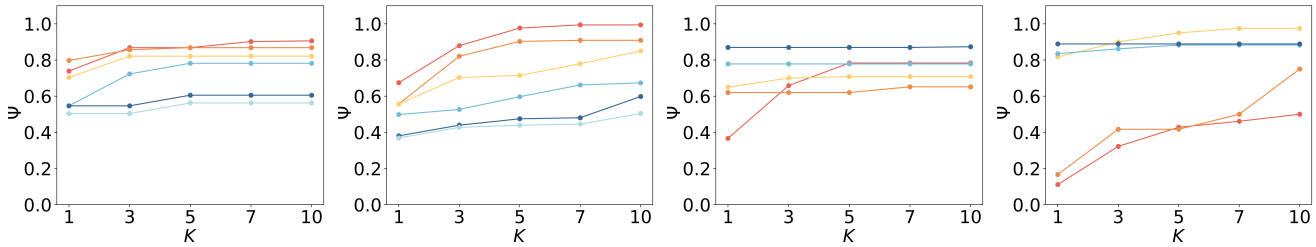
*2) **Attack Precision**:* The experiments of attack precision P are shown in Figs. 7 and 8, where P is discussed under different road network areas, Q^* and K. The y-axes in Figs. 7(a) and 7(b) are the average values of P among different driving trajectory lengths Q^*, while the y-axes in Figs. 7(c) and 7(d) are the average values of P among different road network areas. Unlike the attack success rate Ψ that increases with K, the CAN-Trace attack has a higher values of P with a smaller K as shown in Figs. 7 and 8. As evidenced in Fig. 7, P exhibits superior values for $K = 1$ than for $K \in \{3, 5, 7, 10\}$. This is because the first ranking inferred subgraph candidate is extreme and has either a large or zero number of correctly deduced nodes. As demonstrated in Figs. 7(a) and 7(b), P attains a peak value of 79.75 % in urban when $K = 1$ within a 10 km × 10 km area and a peak value of 67.45 % in suburban $K = 1$ within a 5 km × 5 km road network area. This also proves that the design of the Top-K ranking rule leads to a better attack result and enhances the attack efficiency.

The attack performance is enhanced when the attack performs on a smaller road network size as indicated in Figs. 7(a), 7(b), 8(a) and 8(b). The observed increase of P can be found at 5 km × 5 km road network area compared to the 10 km × 10 km in urban road network, as indicated in Fig. 7(a). As shown in Figs. 7(a) and 7(b), the attack precision P stabilizes after surpassing a road network area value of 20 km × 20 km in the urban region and that of 25 km × 25 km in the suburban region, respectively. Consistent with the trend of Ψ in Figs. 6(c) and 6(d), the attack precision P increases with a longer driving trajectory Q^* as shown in Figs. 7(c) and 7(d). P increases from 36.67 % to 87.04 % in urban region and raises from 11.11 % to 88.89 % in suburban region when Q^* grows from 5 to 15 with $K = 1$. The attack precision P has a significant increase when $Q^* \geq 10$, especially in suburban road networks, as demonstrated in Fig. 7(d).

As shown in Fig. 8, the overall attack performance of P when $K = 1$ is better than that of $K = 3$. Compared the Figs. 8(a) and 8(c), the average value of P is smaller when $K = 3$ than $K = 1$ in urban road networks. The same trend can be found in Figs. 8(b) and 8(d). The reason is that the size of $\hat{R}$ is bigger with a larger K value which decreases the average value of P. However, a larger K can be used to avoid extreme cases as shown in Figs. 8(b) and 8(d) where the CAN-Trace attack has $P = 0$ when $K = 1$ but $P > 0$ when $K = 3$ for the driving trajectory length $Q^* = 7$ in the road network areas of 10 km × 10 km and 15 km × 15 km. In terms of the Q^*, the attack precision tends to increase while the Q^* grows. As indicated in Fig. 8(b), CAN-Track may have no output in some extreme cases, i.e., when $K = 1$ and $Q^* = 7$ for the road network area larger than 5 km × 5 km in suburban region. The same case happens when $K = 3$ and $Q^* = 7$ for the area larger than 15 km × 15 km, as shown in Fig. 8(d). This indicates that the extreme cases can be avoided by enlarging the length of the driving trajectory, i.e., a greater Q^*.

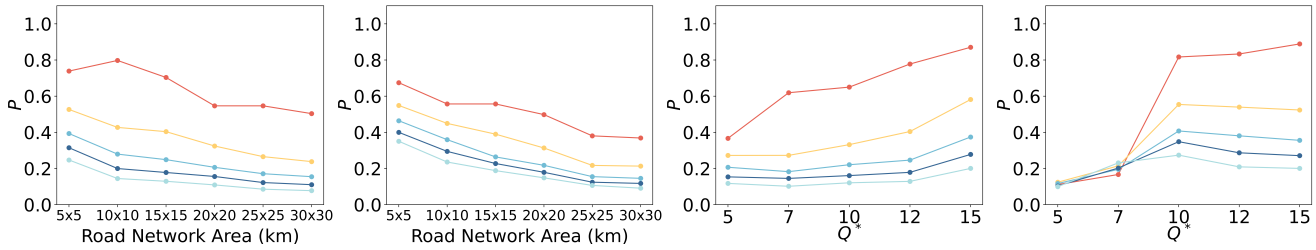
*3) **Spatial Distance Offset**:* The spatial distance offset is compared under different road network areas and Q^* in Fig 9. The attack results filtered by different K are compared among different settings. The y-axes in Figs. 9(a) and 9(b) are the average values of D among different Q^*, and the y-axes in Figs. 9(c) and 9(d) are the average values of D among different road network areas. As shown in Fig. 9, the CAN-Trace attack wins fewer distances offset D in suburban rather than urban road network areas. This is because the suburban region can have a similar road network pattern across the area and far away from each other. The average spatial distance offset D ranges from 0.48 km to 1.30 km in 5 km × 5 km and from

[3]Strict comparisons are challenging due to differences in experimental setups, such as data sources, cities, and trajectories. We have attempted to compare results under similar parameters.



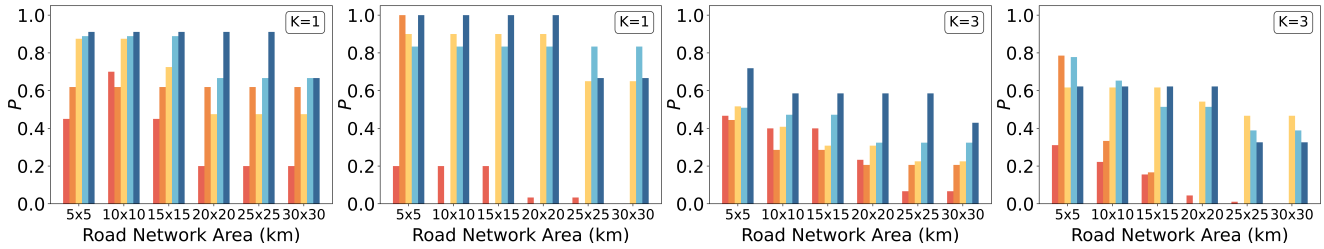
(a) Attack within urban region grouped by road network area (b) Attack within suburban region grouped by road network area (c) Attack within urban region grouped by Q^* (d) Attack within suburban region grouped by Q^*

Fig. 6. Comparison of attack success rate Ψ within urban and suburban road networks: y-axes are the average values of Ψ among different Q^* in subplots (a) and (b), and different road network areas in subplots (c) and (d). The bar colors in subplots (a) and (b) represent the size of road network areas as follows: • 5 km × 5 km, • 10 km × 10 km, • 15 km × 15 km, • 20 km × 20 km, • 25 km × 25 km, and • 30 km × 30 km. The bar colors in subplots (c) and (d) represent the numbers of nodes in the graph of actual driving trajectory as follows: • $Q^* = 5$, • $Q^* = 7$, • $Q^* = 10$, • $Q^* = 12$, and • $Q^* = 15$.



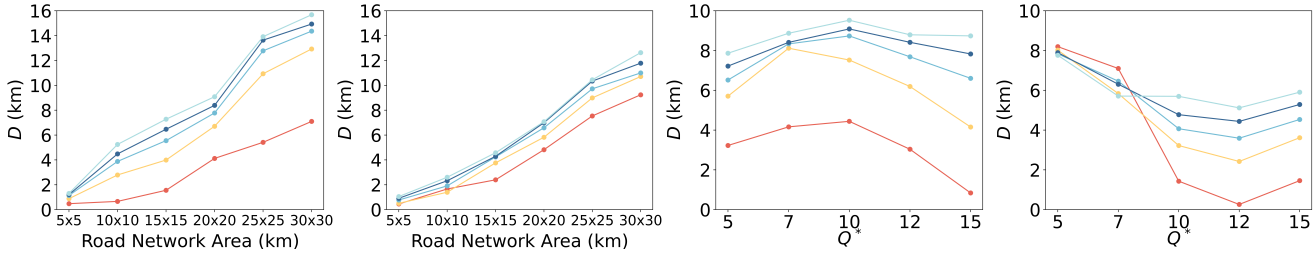
(a) Attack within urban region grouped by K (b) Attack within suburban region grouped by K (c) Attack within urban region grouped by K (d) Attack within suburban region grouped by K

Fig. 7. Comparison of attack precision P within urban and suburban road networks: y-axes are the average values of P among different Q^* in (a) (b) and different road network areas in (c) (d). The bar colors in all subplots represent the values of Top-K as follows: • $K = 1$, • $K = 3$, • $K = 5$, • $K = 7$, and • $K = 10$.



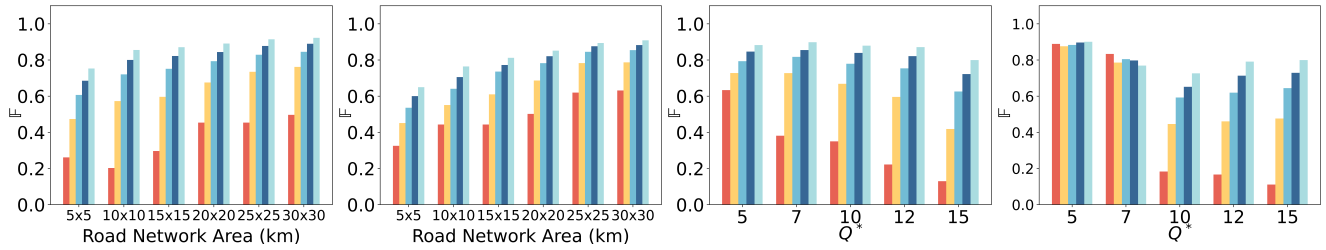
(a) Attack within urban region when $K = 1$ grouped by Q^* (b) Attack within suburban region when $K = 1$ grouped by Q^* (c) Attack within urban region when $K = 3$ grouped by Q^* (d) Attack within suburban region when $K = 3$ grouped by Q^*

Fig. 8. Comparison of attack precision P of $K \in \{1, 3\}$ within urban and suburban road networks: y-axes are the average values of P. The bar colors in all subplots represent the numbers of nodes in the graph of actual driving trajectory as follows: $Q^* = 5$, $Q^* = 7$, $Q^* = 10$, $Q^* = 12$, and $Q^* = 15$.



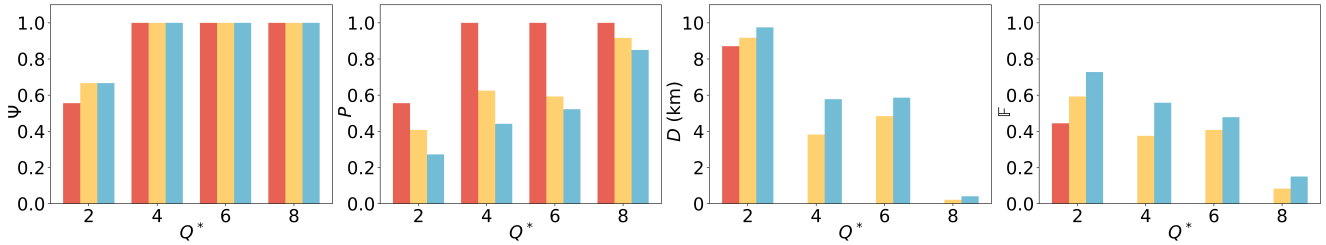
(a) Attack within urban region grouped by K (b) Attack within suburban region grouped by K (c) Attack within urban region grouped by K (d) Attack within suburban region grouped by K

Fig. 9. Comparison of spatial distance offset D within urban and suburban road networks: y-axes are the average values of D among different Q^* in (a)(b) and different road network area in (c)(d). The bar colors in all subplots represent the values of Top-K as follows: • $K = 1$, • $K = 3$, • $K = 5$, • $K = 7$, and • $K = 10$.



(a) Attack within urban region grouped by K (b) Attack within suburban region grouped by K (c) Attack within urban region grouped by K (d) Attack within suburban region grouped by K

Fig. 10. Comparison of false negative rate $\mathbb{F}$ within urban and suburban road networks: y-axes are the average values of $\mathbb{F}$ among different road network areas in (a) (b) and different Q^* in (c) (d). The bar colors in all subplots represent the values of Top-K as follows: $K = 1$, $K = 3$, $K = 5$, $K = 7$, and $K = 10$.



(a) Ψ within remote region grouped by K (b) P within remote region grouped by K (c) D within remote region grouped by K (d) $\mathbb{F}$ within remote region grouped by K

Fig. 11. Evaluation in remote regions: y-axes are the average values of attack success rate Ψ in (a), the average values of the attack precision P in (b), the average values of spatial distance offset D in (c), and the average values of false negative $\mathbb{F}$ in (d). The x-axes are the numbers of nodes in the graph of actual driving trajectories, i.e., Q^*. The road network area is 30 km× 30 km. The bar colors in all subplots represent the values of Top-K as follows: $K = 1$, $K = 3$, and $K = 5$.

7.10 km to 15.68 km in 30 km × 30 km within the urban road network. In the suburban region, D varies from 0.45 km to 1.05 km and from 9.24 km to 12.63 km in the area size of 5 km × 5 km and 30 km × 30 km, respectively. As indicated in Figs. 9(a) and 9(b), the spatial distance offset D has a robust positive correlation with the size of the road network. There is a notable decrease when the road network area becomes smaller, which means that CAN-Trace attack tend to locate the deduced driving trajectory close to the actual driving trajectory. The observed trend can also be attributed to the fact that a smaller road network area yields a smaller number of subgraph candidates. As demonstrated in Figs. 9(c) and 9(d), D decreases with a bigger Q^*. Similar to the attack precision P, the spatial distance offset D gains better attack results when $Q^* \geq 10$, especially in the suburban road networks indicated in Fig. 9(d).

Like the attack precision P, the overall attack performance of D is enhanced with a smaller K. Notably, D for $K = 1$ is about half that for $K = 3$ as demonstrated in Fig. 9. The best performance of D comes with the lowest value 0.8394 km with $K = 1$ and is 4.15 km when $K = 3$ in urban road networks, as shown in Fig. 9(c). In suburban road network, D drops to 0.26 km when $K = 1$ but is greater than 2.42 km when $K \in \{3, 5, 7, 10\}$ as demonstrated in Fig. 9(d). The difference of D between $K = 1$ and $K \in \{3, 5, 7, 10\}$ is extremely large, as shown in Figs. 9(c) and 9(d). The reason is that a larger K value brings in candidates with different node matches, and some are less aligned.

*4) **False Negative Rate**:* The attack false negative rate $\mathbb{F}$ is examined in Fig. 10 across different road network areas, the number of actual driving trajectories, and K for Top-K, in urban and suburban regions. The false negative rate can be high, especially when K is large, e.g., $K = 7$. This is because the Top-K selection mechanism includes multiple trajectories even though only one is correct, increasing the numbers of incorrect trajectories and nodes selected. For example, in the case of $Q^* = 15$, the false negative rate $\mathbb{F}$ is 11.11% when $K = 1$ in suburban region, as shown in Fig. 10(d), indicating that CAN-Trace has identified majority nodes in the actual trajectory. However, as K increases, incorrect trajectories are included in the Top-K selection, leading to a higher $\mathbb{F}$, e.g., $\mathbb{F} = 79.96\%$ for $K = 10$, as the second through tenth results barely cover any right nodes.

The false negative rate $\mathbb{F}$ gradually increases as the road network size expands. For example, when $K = 1$, $\mathbb{F}$ is 32.55% for the 5 km × 5 km road networks in suburban regions and 53.14% for the 30 km × 30 km road networks in suburban regions, as shown in Fig. 10(b). The false negative rate $\mathbb{F}$ decrease as the number of nodes in the actual driving trajectories increases. According to Fig. 10(c), when $K = 1$, $\mathbb{F}$ is 63.33% on average for all trajectory graphs with five nodes and drops to only 12.96% for all trajectory graphs with 15 nodes. The experiment results reveal that CAN-Trace requires a certain number of nodes to accurately identify the actual trajectories, and K should be tuned to balance the attack success rate and the false negative rate.

*5) **Performance in Remote Regions**:* The proposed CAN-Trace attack is also validated in remote regions. The attack performance metrics, including success rate (Ψ), precision (P), spatial distance offset (D), and false negative rate ($\mathbb{F}$), are illustrated in Fig. 11 Testing results confirm the effectiveness of the CAN-Trace attack in remote regions. Similar to the results in urban and suburban regions, the attack performance improves with increasing Q^*, leading to higher Ψ and P and lower D and $\mathbb{F}$. Notably, the results in the remote regions demonstrate better performance of the proposed attack compared to urban and suburban regions, with the attack success rate (Ψ) reaching the upper bound with a small K and few nodes in the driving trajectory graph. This is due to the strong heterogeneity of road segments in remote regions, which allows for exclusive matching.

D. Discussion

1) Lesson Learned: In summary, the proposed CAN-Trace attack has a good performance on the attack success rate Ψ, attack precision P, and spatial distance offset D in the real-world environment. In our experiments, the CAN-Trace attack infers the vehicle driving trajectory with a higher value of Ψ and P within a smaller road network area, which means the attack is quite efficient in a small size of the road network area. CAN-Trace attack performance in the suburban region is slightly superior to those in the urban region, especially when the driving trajectory length Q^* is greater than 10. The experiments highlight the impact of the Top-K value, which wins the highest attack precision P when $K = 1$ and gains a good attack success rate Ψ representing the coverage when K is greater than 5. Thus, a K value between 1 and 5 is recommended to balance the attack efficiency and coverage.

2) Privacy Concern: The proposed CAN-Trace attack can persist because it utilizes basic vehicle motion data, i.e., speed and pedal data, from the standard OBD-II protocol that is widely used in the automotive industry. The attack could compromise drivers' privacy on personal addresses, locations, and driving trajectories by analyzing CAN messages alone. In our experiments, all driving data are collected with consent. In practice, the application of the proposed CAN-Trace should follow privacy regulations such as the General Data Protection Regulation (GDPR) in Europe and the California Consumer Privacy Act (CCPA). Organizations like car rental or logistics companies could leverage CAN-Trace to track vehicles, provided there is full disclosure and compliance with these regulations.

3) Mitigation: To safeguard against the proposed CAN-Trace attack, we recommend actions from car manufacturers, drivers, and CAN-based service providers. Car manufacturers should design secure, privacy-preserving CAN networks with secure access controls. Drivers should regularly inspect the OBD-II port and other potential CAN sniffers to prevent unauthorized physical access. CAN-based service providers should recognize the privacy risks of disclosing CAN messages and implement encryption, data anonymization (e.g., k-anonymity), and privacy-preserving technologies (e.g., differential privacy) during the storage, processing, and sharing of CAN data. In summary, all stakeholders must recognize the privacy risks of trajectory leakage from CAN messages, ensure that only trusted parties have access, and apply anonymity, data privacy, and other privacy-preserving technologies to protect CAN data.

V. Related Work

A. Map-Matching: Methods and Sensor Utilization

The existing map-matching methods [36] that detect vehicle location and trajectory are discussed in this section, focusing on the comparison of data sources. Map-matching is to find the road segments on which the vehicle drives and the location of the vehicles to finally map out the driving trajectory. Since the driving trajectory contains the personal information [37], the sensitive information such as the driver's habits and identity can be inferred from the trajectory dataset [38], [39], which leads to the driver's privacy leakage. To infer the locations and trajectories of the vehicle, adversaries can launch side-channel attacks with the Global Positioning System (GPS) data [18], [40], passive sensor data [17], [21], [41]–[43], and the vehicular network data [44]–[46].

In [43], the vehicle motion data from mobile magnetometer sensors is introduced as additional data in map-matching. In [21], Li et al. revealed the driving trajectory by matching intersection angles with the magnetometer sensor data. Li et al. developed the first attack model [21] to match car turn angles with the road network intersection angles to construct the driving trajectory. The GPS data is not required in the developed attack model, but the compass data from the mobile phone. Unfortunately, the attack model degrades the performance in the case of disorientation due to the perturbed phone position by drivers or passengers.

B. GPS Data Utilization

The most used data to deduce the vehicle trajectories is the GPS data. In [18], the authors utilized a passive GPS device to collect GPS data and vehicle state by inferring the victims' home and working addresses from a large number of guesses. The map-matching process using GPS data is to convert a sequence of GPS data into a sequence of road segments [47]. However, the GPS data is not reliable due to the difficult access and data noise and loss. The missing data caused by the GPS outage problem require additional data such as the odometer, lidar or camera data [13]–[17], [48]–[51] to enhance the inference accuracy.

C. Innovative Approaches to Trajectory Reconstruction

Unlike identifying only a few road points, Guhu et al. analyzed the sequence of vehicle motions (i.e., movement, stop, turn) collected by a small wireless tag to reconstruct the complete driving path with knowing the initial and final GPS position [40]. Xiao et al. classified the road section types with the motion information from the OBD data and integrated the road section types with GPS data to reconstruct the driving trajectory [17]. The Gated Recurrent Unit (GRU) model is used to identify the candidate path during GPS

TABLE II
COMPARISON WITH TRAJECTORY DETECTION STUDIES

Reference	Data Sources	Approaches	Advantages	Limitations
L2MM [33]	GPS data	Deep Learning	Handel poor-quality GPS data	GPS coverage and access permission Data quality still matters
GOI [20]	GPS data Vehicle motion data	SVR	Data fusion for robustness	GPS coverage and access permission Specific vehicle motion data
DMM [34]	Cell tower location data Cell access data	RNN	GPS-agnostic Passive interference by operators	Cellular network coverage Cell towers locations
MBT [21]	Magnetometer data	Turn angle matching	GPS-agnostic	Need external fixed IMU High cost to calculate turn angles
DaRoute [32]	IMU data	Route ranking	GPS-agnostic No wiring to vehicles	Need external fixed IMU Limited IMU data accuracy
Invasion [35]	Mobile motion data	Noise removing DTW matching	GPS-agnostic Tolerate human-generate noise	Need external mobile/IMU Restrict to subway trajectories
CAN-Trace	CAN data (speed and pedal)	Subgraph matching	GPS-agnostic No attached motion sensors Generic vehicle motion data	Need access to CAN messages

GPS, SVR, RNN, IMU, DTW and CAN stand for Global Positioning System, Support Vector Regression, Recurrent Neural Network, Inertial Measurement Unit, Dynamic Time Warping, and Controller Area Network, respectively.

outages. However, both approaches solve the GPS outage problem that only constructs partial driving trajectory and requires GPS data.

D. Challenges and Novel Attack Surfaces

The GPS data or sensor data from external devices (e.g., smartphones) are unreliable data sources due to issues like restricted access and data noise. The GPS outage problem can be partially addressed by incorporating additional data, such as vehicle motion data from compass sensors. The vehicle motion data collected from the smartphone is easily perturbed by unexpected user movement and error positioning [17]. In order to enhance attack efficiency, various attack surfaces are being investigated, including tire pressure sensors [42] and the hardware-embedded scrambling algorithm [46]. However, these attack surfaces either demand sophisticated techniques or involve complex access requirements.

E. Comparison Highlights

Various data sources have been studied for driving trajectory detection. GPS data enables accurate position and trajectory detection [20], [33], although it suffers from limited coverage and requires sensitive permissions to access. In the absence of GPS, DMM [34] leverages cell connectivity data and cell tower locations to infer the trajectory of mobile users; however, this approach is suitable only for attackers with extensive knowledge of mobile networks. Motion data has also gained interest for trajectory detection [21], [32], but it requires an external Inertial Measurement Unit (IMU) to be mounted in vehicles, with the IMU remaining fixed during driving to minimize human-generated noise. In [35], the authors designed a new algorithm to remove human-generated noise but only manage to match subway trajectories of passengers.

This paper proposes a new vector for driving trajectory detection by leveraging the vehicle motion data from in-vehicle Controller Area Network (CAN), specifically speed and pedal data. These vehicle-generated CAN data can be more accurate than the data from external IMU and do not require the external IMU. The CAN data can be collected via the OBD-II ports of vehicles or obtained from CAN data-based service providers.

VI. CONCLUSION AND FUTURE WORK

In this paper, we proposed the CAN-Trace attack to deduce the vehicle driving trajectory using geolocation data in real-world road networks. This work is the first to infer driving trajectories using CAN messages and to apply subgraph matching to align the trajectories with road networks. The proposed CAN-Trace attack obtains the vehicle kinematic data by accessing the CAN bus via the OBD-II port, ensuring the integrity of the data and the stealthiness of the proposed attack. The proposed attack uses the Top-K ranking rule to improve the attack efficiency. The proposed CAN-Trace attack is evaluated with real-world driving data and traffic scenarios. As demonstrated by the experimental results, the CAN-Trace attack performs effectively in both urban and suburban regions.

In future work, we plan to consider weather conditions, traffic patterns, city layouts, vehicle-specific factors, and driving behaviors to further refine and enhance the method. Additionally, we aim to develop new algorithms capable of handling pure CAN messages or integrating CAN data with other sources to effectively and accurately reconstruct driving trajectories.

ACKNOWLEDGMENT

We thank our industry partner IAG for providing access to test vehicles. We thank Thanh Phuoc Nguyen for assisting with technical issues and experiments. This work is funded in part by the University of Technology Sydney, Insurance Australia Group (IAG), and the iMOVE CRC under Grant 5-028. This work is also supported by the Cooperative Research Centres program, an Australian Government initiative.

REFERENCES

[1] J. Krumm, "A survey of computational location privacy," *Pers. Ubiquitous Comput.*, vol. 13, pp. 391–399, 2009.

[2] X. Wang, T. Gu *et al.*, "A user study on the capability of three geo-based features in analyzing and locating trajectories," *IEEE Trans. Intell. Transp. Syst.*, vol. 20, no. 9, pp. 3375–3385, 2018.

[3] Z. Ma, F. Kargl, and M. Weber, "A location privacy metric for V2X communication systems," in *2009 IEEE Sarnoff Symposium*. IEEE, 2009, pp. 1–6.

[4] J. Huang, D. Fang, Y. Qian, and R. Q. Hu, "Recent advances and challenges in security and privacy for V2X communications," *IEEE Open J. Veh. Technol.*, vol. 1, pp. 244–266, 2020.

[5] B. Ma, X. Wang *et al.*, "Location privacy threats and protections in future vehicular networks: A comprehensive review," *arXiv preprint arXiv:2305.04503*, 2023.

[6] G. P. Corser, H. Fu, and A. Banihani, "Evaluating location privacy in vehicular communications and applications," *IEEE Trans. Intell. Transp. Syst.*, vol. 17, no. 9, pp. 2658–2667, 2016.

[7] B. Ma, X. Wang, W. Ni, and R. P. Liu, "Personalized location privacy with road network-indistinguishability," *IEEE Trans. Intell. Transp. Syst.*, vol. 23, no. 11, pp. 20 860–20 872, 2022.

[8] E. J. Krakiwsky, C. B. Harris, and R. V. Wong, "A Kalman filter for integrating dead reckoning, map matching and GPS positioning," in *IEEE PLANS'88., Position Location and Navigation Symposium, Record.'Navigation into the 21st Century'*. IEEE, 1988, pp. 39–46.

[9] R. Assam and T. Seidl, "Private map matching: Realistic private route cognition on road networks," in *2013 IEEE 10th International Conference on Ubiquitous Intelligence and Computing and 2013 IEEE 10th International Conference on Autonomic and Trusted Computing*. IEEE, 2013, pp. 178–185.

[10] J. M. Carter and A. E. Ferber, "Using map matching for deidentification of connected vehicle locations," *IEEE Consum. Electron. Mag.*, vol. 8, no. 6, pp. 111–116, 2019.

[11] H.-K. Kong, M. K. Hong, and T.-S. Kim, "Security risk assessment framework for smart car using the attack tree analysis," *J. Ambient Intell. Humaniz. Comput.*, vol. 9, no. 3, pp. 531–551, 2018.

[12] D. Jianjun and C. Xiaohong, "An improved map-matching model for GPS data with low polling rates to track vehicles," in *Proceedings 2011 International Conference on Transportation, Mechanical, and Electrical Engineering (TMEE)*. IEEE, 2011, pp. 299–302.

[13] R. Roncella, F. Remondino, and G. Forlani, "Photogrammetric bridging of GPS outages in mobile mapping," in *Videometrics VIII*, vol. 5665. SPIE, 2005, pp. 308–319.

[14] J. Liu and G. Guo, "Vehicle localization during GPS outages with extended Kalman filter and deep learning," *IEEE Trans. Instrum. Meas.*, vol. 70, pp. 1–10, 2021.

[15] J. Georgy, T. Karamat, U. Iqbal, and A. Noureldin, "Enhanced MEMS-IMU/Odometer/GPS integration using mixture particle filter," *GPS Solut.*, vol. 15, pp. 239–252, 2011.

[16] S. Sasani, J. Asgari, and A. Amiri-Simkooei, "Improving MEMS-IMU/GPS integrated systems for land vehicle navigation applications," *GPS Solut.*, vol. 20, pp. 89–100, 2016.

[17] C. Chen, Y. Ding *et al.*, "Trajcompressor: An online map-matching-based trajectory compression framework leveraging vehicle heading direction and change," *IEEE Trans. Intell. Transp. Syst.*, vol. 21, no. 5, pp. 2012–2028, 2019.

[18] M. U. Iqbal and S. Lim, "Privacy implications of automated GPS tracking and profiling," *IEEE Technol. Soc. Mag.*, vol. 29, no. 2, pp. 39–46, 2010.

[19] J. Lim, K. H. Choi *et al.*, "Land vehicle positioning in urban area by integrated GPS/BeiDou/OBD-II/MEMS IMU," in *2016 IEEE International Conference on Intelligent Transportation Engineering (ICITE)*. IEEE, 2016, pp. 176–180.

[20] Z. Xiao, P. Li *et al.*, "GOI: A novel design for vehicle positioning and trajectory prediction under urban environments," *IEEE Sens. J.*, vol. 18, no. 13, pp. 5586–5594, 2018.

[21] Z. Li, Q. Pei *et al.*, "Location privacy violation via GPS-agnostic smart phone car tracking," *IEEE Trans. Veh. Technol.*, vol. 67, no. 6, pp. 5042–5053, 2018.

[22] S. C. HPL, "Introduction to the Controller Area Network (CAN)," *Application Report SLOA101*, pp. 1–17, 2002.

[23] W. Voss, *A comprehensible guide to Controller Area Network*. Copperhill Media, 2008.

[24] K. H. Johansson, M. Törngren, and L. Nielsen, "Vehicle applications of Controller Area Network," *Handbook of Networked and Embedded Control Systems*, pp. 741–765, 2005.

[25] A. Szijj, L. Buttyán, and Z. Szalay, "Hacking cars in the style of Stuxnet," *Laboratory of Cryptography and System Security, Budapest, Hungary*, 2015.

[26] W. Yan, "A two-year survey on security challenges in automotive threat landscape," in *2015 International Conference on Connected Vehicles and Expo (ICCVE)*. IEEE, 2015, pp. 185–189.

[27] D. Klinedinst and C. King, "On Board Diagnostics: Risks and vulnerabilities of the connected vehicle," *CERT Coordination Center, Tech. Rep*, 2016.

[28] H. Wen, Q. A. Chen *et al.*, "Plug-N-Pwned: Comprehensive vulnerability analysis of OBD-II dongles as a new Over-the-Air attack surface in automotive IoT," in *29th USENIX Security Symposium (USENIX Security 20)*, 2020, pp. 949–965.

[29] X. Lin, B. Ma *et al.*, "Multi-layer reverse engineering system for vehicular Controller Area Network messages," in *2022 IEEE 25th International Conference on Computer Supported Cooperative Work in Design (CSCWD)*. IEEE, 2022, pp. 1185–1190.

[30] L. P. Cordella, P. Foggia, C. Sansone, and M. Vento, "A (sub) graph isomorphism algorithm for matching large graphs," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 26, no. 10, pp. 1367–1372, 2004.

[31] D. Cheng, J. Huang, S. Zhang, S. Xia, G. Wang, and J. Xie, "K-Means clustering with natural density peaks for discovering arbitrary-shaped clusters," *IEEE Transactions on Neural Networks and Learning Systems*, vol. 35, no. 8, pp. 11 077–11 090, 2024.

[32] C. Roth, N. T. Dinh, M. Roßberger, and D. Kesdoğan, "DaRoute: Inferring trajectories from zero-permission smartphone sensors," in *2021 18th International Conference on Privacy, Security and Trust (PST)*. IEEE, 2021, pp. 1–10.

[33] L. Jiang, C.-X. Chen, and C. Chen, "L2MM: learning to map matching with deep models for low-quality GPS trajectory data," *ACM Transactions on Knowledge Discovery from Data*, vol. 17, no. 3, pp. 1–25, 2023.

[34] Z. Shen, K. Yang, X. Zhao, J. Zou, W. Du, and J. Wu, "DMM: A deep reinforcement learning based map matching framework for cellular data," *IEEE Transactions on Knowledge and Data Engineering*, 2024.

[35] H. Kim, Y. Jeon, and J. W. Yoon, "Invasion of location privacy using online map services and smartphone sensors," in *Proceedings of the 2023 ACM Asia Conference on Computer and Communications Security*, 2023, pp. 41–52.

[36] M. A. Quddus, W. Y. Ochieng, and R. B. Noland, "Current map-matching algorithms for transport applications: State-of-the art and future research directions," *Transp. Res. C: Emerg. Technol.*, vol. 15, no. 5, pp. 312–328, 2007.

[37] Y. Xu, C. X. Wang, Y. Song, and W. P. Tay, "Preserving trajectory privacy in driving data release," in *ICASSP 2022-2022 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE, 2022, pp. 3099–3103.

[38] T. Ma and F. Song, "A trajectory privacy protection method based on random sampling differential privacy," *ISPRS Int. J. Geo-Inf.*, vol. 10, no. 7, p. 454, 2021.

[39] Y. Xin, Z.-Q. Xie, and J. Yang, "The privacy preserving method for dynamic trajectory releasing based on adaptive clustering," *Inf. Sci.*, vol. 378, pp. 131–143, 2017.

[40] S. Guha, K. Plarre *et al.*, "AutoWitness: Locating and tracking stolen property while tolerating GPS and radio outages," *ACM Trans. Sens.*, vol. 8, no. 4, pp. 1–28, 2012.

[41] B.-J. Ho, P. Martin, P. Swaminathan, and M. Srivastava, "From pressure to path: Barometer-based vehicle tracking," in *Proceedings of the 2nd ACM International Conference on Embedded Systems for Energy-Efficient Built Environments*, 2015, pp. 65–74.

[42] I. Rouf, R. Miller *et al.*, "Security and privacy vulnerabilities of In-Car wireless networks: A tire pressure monitoring system case study," in *19th USENIX Security Symposium (USENIX Security 10)*, 2010.

[43] J. Han, E. Owusu *et al.*, "Accomplice: Location inference using accelerometers on smartphones," in *2012 Fourth International Conference on Communication Systems and Networks (COMSNETS 2012)*. IEEE, 2012, pp. 1–9.

[44] M. Raya and J.-P. Hubaux, "Securing vehicular Ad hoc networks," *J. Comput. Secur.*, vol. 15, no. 1, pp. 39–68, 2007.

[45] L. B. Othmane, H. Weffers *et al.*, "A survey of security and privacy in connected vehicles," in *Wireless sensor and mobile ad-hoc networks*. Springer, 2015, pp. 217–247.

[46] B. Bloessl, C. Sommer *et al.*, "The scrambler attack: A robust physical layer attack on location privacy in vehicular networks," in *2015 International Conference on Computing, Networking and Communications (ICNC)*. IEEE, 2015, pp. 395–400.

[47] L. Cao and J. Krumm, "From GPS traces to a routable road map," in *Proceedings of the 17th ACM SIGSPATIAL International Conference on Advances in Geographic Information Systems*, 2009, pp. 3–12.

[48] M. Aftatah, A. Lahrech, and A. Abounada, "Fusion of GPS/INS/Odometer measurements for land vehicle navigation with GPS outage," in *2016 2nd International Conference on Cloud Computing Technologies and Applications (CloudTech)*. IEEE, 2016, pp. 48–55.

[49] G. Huang, "Visual-inertial navigation: A concise review," in *2019 International Conference on Robotics and Automation (ICRA)*. IEEE, 2019, pp. 9572–9582.

[50] S. Zhao, H. Zhang *et al.*, "Super odometry: IMU-centric LiDAR-visual-inertial estimator for challenging environments," in *2021 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE, 2021, pp. 8729–8736.

[51] Y. Liu, Q. Luo, and Y. Zhou, "Deep learning-enabled fusion to bridge GPS outages for INS/GPS integrated navigation," *IEEE Sens. J.*, vol. 22, no. 9, pp. 8974–8985, 2022.