Securing LiDAR Communication through Watermark-based Tampering Detection

Michele Marazzi, Stefano Longari, Michele Carminati, and Stefano Zanero
Politecnico di Milano
michele1.marazzi@mail.polimi.it stefano.longari@polimi.it, michele.carminati@polimi.it, stefano.zanero@polimi.it

***Abstract*—With the increasing interest in autonomous vehicles (AVs), ensuring their safety and security is becoming crucial. The introduction of advanced features has increased the need for various interfaces to communicate with the external world, creating new potential attack vectors that attackers can exploit to alter sensor data. LiDAR sensors are widely employed to support autonomous driving features and generate point cloud data used by ADAS to 3D map the vehicle's surroundings. Tampering attacks on LiDAR-generated data can compromise the vehicle's functionalities and seriously threaten passengers and other road users. Existing approaches to LiDAR data tampering detection show security flaws and can be bypassed by attackers through design vulnerabilities. This paper proposes a novel approach for tampering detection of LiDAR-generated data in AVs, employing a watermarking technique. We validate our approach through experiments to prove its feasibility in real-world time-constrained scenarios and its efficacy in detecting LiDAR tampering attacks. Our approach performs better when compared to the current state-of-the-art LiDAR watermarking techniques while addressing critical issues related to watermark security and imperceptibility.**

I. Introduction

Autonomous driving technology refers to the partial or complete replacement of human driving activity with an automated system based on electronic and mechanical devices. With advancements in technology, new commercially available vehicles are becoming increasingly rich in terms of driver assistance features. These features often require different interfaces to communicate with the external world, potentially providing new entry points for the attackers. The implementation of autonomous driving technology can deliver numerous benefits to society, including improved safety, reduced traffic congestion, enhanced vehicle management, and increased transport availability for individuals unable to drive [10]. Safety is a crucial aspect of autonomous driving, as the system must make accurate decisions in every situation to ensure the safety of both the driver and the environment. To make such decisions, vehicles are required to obtain precise and various data on their surroundings. LiDAR sensors, using laser beams to generate a point cloud representation of the surrounding environment, are considered essential for generating effective and precise data for autonomous driving. LiDAR data are combined with data from other sensors, such as cameras [27] in Advanced Driver Assistance Systems (ADAS) to provide a comprehensive understanding of the environment and control the vehicle's actions. The advantages of LiDAR point clouds in relation to camera images, however, are multiple, from depth perception to darkness visibility and weather dependency. The transmission of data from the LiDAR Electronic Control Unit (ECU) to the ADAS depends on the on-board networks of the vehicle, which lack integrated security measures and have been the target of many attack demonstrations [4], [16], [8] in the last decade. Attackers can inject frames into the network, spoofing the transmitted LiDAR data by inserting or removing points in the point cloud, potentially taking control of the Autonomous Vehicle (AV) actions.

To mitigate this issue, this research work aims to develop a solution to verify the integrity of LiDAR-generated point clouds to ensure that the ADAS does not act on tampered information. The proposed solution is based on watermarking, which utilizes the imperceptible displacement of selected points in the LiDAR scan to conceal a secret message. This message is then extracted and verified at the receiver side, providing a lightweight integrity check that can be performed in parallel with the inference operations on the scan and can recognize the areas of the point cloud under attack.

The results on attacks injected on the widely-exploited KITTI Dataset [11] indicate that the proposed approach has high efficacy in detecting and locating manipulations on the LiDAR-generated point clouds while respecting time constraints and applying imperceptible modifications that do not impact the inferential procedures in ADAS.

Our main contributions are the following:

- A publicly available[1] novel approach for detecting attacks against LiDAR point cloud scans, based on fragile watermarking.
- An evaluation of the proposed approach to prove its feasibility in real-world time-constrained scenarios and its efficacy in detecting LiDAR tampering attacks.
- A comparison with state-of-the-art techniques for LiDAR tampering detection based on watermarking, highlighting their limitations and shortcomings.

II. Primer on watermarking

Data hiding techniques are a set of security measures that are used to protect confidential information from unauthorized

Symposium on Vehicles Security and Privacy (VehicleSec) 2024
26 February 2024, San Diego, CA, USA
ISBN 979-8-9894372-7-6
https://dx.doi.org/10.14722/vehiclesec.2024.23055
www.ndss-symposium.org

[1]The implementation of our approach, alongside all of our experiments, are available online at https://github.com/necst/LiDAR_watermarking.

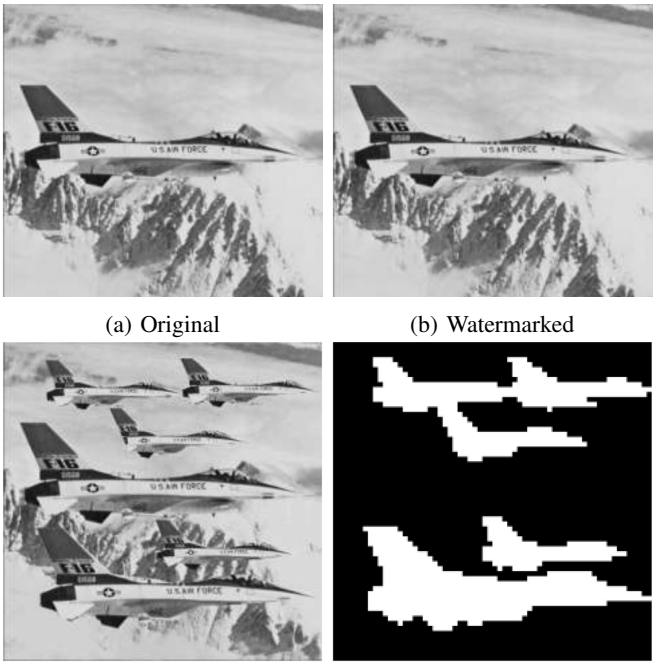
access or alteration. These techniques involve concealing information inside data with minimal modifications to their features. The adjustments made in the data must be imperceptible in the expected usage of the data, where the modified data has to provide the same information as the original. The receiver should be capable of extracting hidden information and utilizing it while performing necessary operations on data. While there are multiple categories of data hiding techniques [24], namely steganography, reversible data hiding, and watermarking, we focus on the latter, the only one of the three where the final goal is not to hide the information. Watermarking conceals information inside data, but the peculiarity is that the hidden message is known by the sender and the receiver and is used by the receiver to detect any modifications made to the shared multimedia data. Specifically, the receiver extracts the secret message from the multimedia and compares it to the known message. Without discrepancies, the receiver can infer that data have not been altered. If discrepancies are found, the receiver can affirm that the data are not the same as the original ones and may further identify the specific elements where the two versions do not match. The primary goal of the communication remains the multimedia data itself, with the secret message serving solely to safeguard its integrity.

Watermarking is widely used nowadays for various applications, such as copyright protection, tampering detection, metadata insertion, or media authentication [25]. A watermarking technique comprises two essential phases: embedding and extraction. The embedding phase, which is executed on the sender side, involves making minor alterations to the exchanged multimedia data to conceal secret information. Each modification conceals a fragment of information, which unveils the complete secret when combined with others. On the receiver side, the extraction phase deciphers the hidden information by associating specific features in the multimedia data with the meaning assigned during the embedding phase. These fragments of information are then merged to disclose the complete secret.

Compared to encryption, watermarking provides various advantages. First, watermarking enables parallel data processing, as the integrity check can be performed concurrently with other necessary operations. In contrast, encryption necessitates a decryption step before any operations can be carried out, which may be time-consuming and impact performance. Secondly, watermarking can locate specific regions within the data that have been tampered with during an attack. This facilitates targeted solutions, preserving the integrity of the remaining data. In contrast, encryption often results in complete data loss when tampering is detected. While Message Authentication Codes (MACs) partially solve the disadvantages of a full encryption process (albeit not being able to identify the region under attack), watermarks are embedded within the data and do not require additional storage or bandwidth to be stored and shared, both potentially limited resources in automotive environments.

A. Watermarking Properties

Watermarking is a versatile technique that can be applied to various scenarios, owing to the various properties that make it suitable for different purposes. The main properties can be



(a) Original (b) Watermarked

(c) Tampered (d) Detected tampered areas

Fig. 1: Example of a tampering detection process based on fragile watermark for digital images from [22].

divided in [20], [9], [25] *robustness* (or *fragility*), *blindness*, *imperceptibility*, and *visibility*.

1) Robustness or Fragility: Digital watermarks can be classified as either robust or fragile. Fragile watermarks are particularly useful for content authentication, as they are designed to detect any intentional or unintentional modification made to transmitted data. They are called ”fragile” because of their high susceptibility to data alteration, as small modifications made to the original data can fully or partially break the watermark. This characteristic is particularly relevant to our research scope. An example of a fragile watermarking approach applied to images is illustrated in Figure 1.

On the contrary, when applied to images, ”robust” watermarks can withstand various modifications that may occur during distribution, such as cropping or compression. Compared to fragile, robust watermarks are challenging to remove without significantly altering the data quality. They are commonly employed in multimedia content to authenticate its origin and ownership. These watermarks are typically invisible to the human eye and can only be retrieved using specific watermarking software.

2) Blindness: Blindness refers to the process of extracting hidden information from watermarked multimedia data. The type of watermark used can be categorized as blind, non-blind, or semi-blind depending on the additional information required by the receiver to extract the hidden data accurately. In the case of a non-blind watermark, the receiver must have access to the original multimedia data in order to extract the watermark successfully. On the other hand, a blind watermark does not require any information about the original data to extract the watermark. A semi-blind watermark combines elements of both blind and non-blind techniques, where the receiver has partial access to the original data. This may include

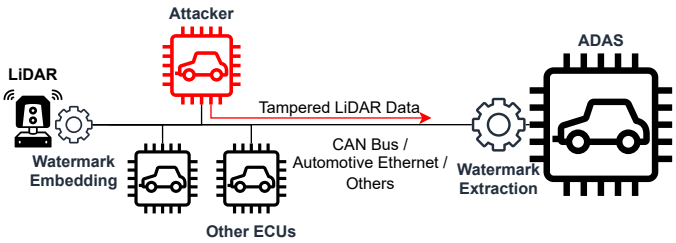


Fig. 2: Overview of the approach interaction with the attacker and vehicle on-board network.

partial information about certain regions of the image or some statistical properties of data.

3) Imperceptibility: Imperceptibility in watermarking refers to the ability of the watermark to introduce a minimal level of distortion to the original data. The marked data should be similar enough to the original data that any operation performed on the two will result in nearly identical outcomes. The importance of imperceptibility lies in the need to ensure that the marked data is not significantly altered or distorted, as this can compromise the quality and integrity of the original content.

4) Visibility: A watermark can be classified as visible or invisible based on its detectability in the marked data. If the watermark is easily identifiable in the marked data, such as a company logo overlaid on a video, then it is considered visible. On the other hand, if the watermark cannot be easily detected in the marked data, it is classified as invisible.

III. Approach

Our goal is to develop a methodology to ensure the integrity of LiDAR data point clouds in real-time contexts that require rapid or parallel processing of information while being able to detect which elements of the point cloud have been tampered with. The proposed methodology involves a technique of fragile watermarking. The LiDAR ECU embeds a hidden message within the generated point cloud and transmits the resulting marked point cloud to ADAS. ADAS then extracts the message from the received point cloud and compares the extracted information with the known one, as both entities share the same information generation technique. In the event of any discrepancy, ADAS can detect the specific area that has been tampered with.

A. Threat model

We consider an attacker capable of feeding spoofed LiDAR point clouds to the receiving device, which is the ADAS in the context of autonomous driving. In the context of autonomous driving, we assume that the attacker has already gained access to the on-board network and can communicate with the ADAS as if it was the LiDAR ECU. Multiple works in the state of the art discuss the feasibility of obtaining access to an on-board ECU, both through physical interaction [4], [15] and remotely [16], and another vast amount of research has been done on studying injection and masquerade attacks on automotive on-board networks [17], [8]. Specifically, we distinguish the attacks that can be performed in the following main categories:

- *Injection attack*: the attacker generates a false point cloud and feeds it to the ADAS.
- *Replay attack*: the attacker collects one or more point clouds sent by the LiDAR ECU and replays them in a different time slot.
- *Object insertion attack*: the attacker intercepts a LiDAR point cloud and inserts points representing a custom element before injecting it. Note that the attacker can ideally use an object obtained by a previous message.
- *Object removal attack*: the attacker intercepts a LiDAR point cloud and removes points representing an element before injecting it.

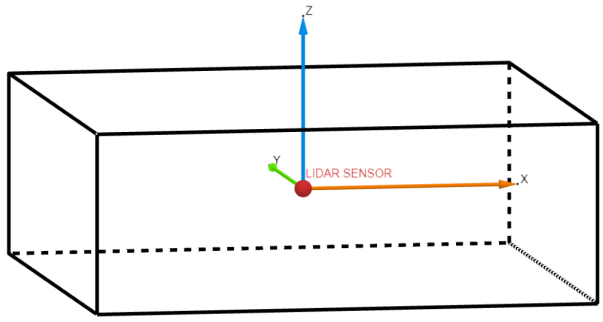
It is important to clarify that we do not consider an attacker whose sole goal is disrupting the communication between the LiDAR and the ADAS since watermarking does not address DoS and similar attacks. We also assume that the attacker's knowledge of the inference process of ADAS enables them to manipulate scans by adding or removing elements in a way that can influence the object detection results.

B. Information generation

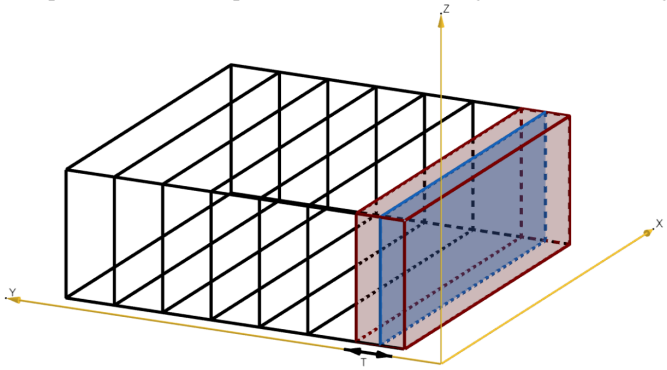
While not being the core focus of our work, the process of generating information to hide is a critical aspect of the security of the watermark. Given the attacks discussed in Section III-A, the hidden message has to be non-predictable to avoid spoofing, and dynamic to avoid replay or insertion attacks. A known solution in the state-of-the-art is the Time-based One-Time Password (TOTP) algorithm [19], a variant of the Hash-based One-Time Password (HOTP) by M'Raihi et al. [18]. The algorithms employ a cryptographic hash function fed with a key and a counter to generate a sequence of pseudo-random numbers. The algorithm's security is based on the security of the hash function used to generate the pseudo-random numbers and the secrecy of the key, which must be known and stored confidentially by all the communicating parties. While in the HOTP algorithm, the dynamicity of the generated hidden message is ensured by the use of the counter, which must be synchronized between communication parties, in the TOTP algorithm, the counter value used to feed the hash function along with the key is replaced with a value that is dependent on the current timestamp. To ensure synchronization between the sender and receiver, the clock of both parties has to be synchronized, and the receiver has to know when to calculate the key to use with the point cloud received at each timestamp. We consider a time-step parameter that should be defined at design time, which is known by each party and is used to determine when to generate a new hidden message. Ideally, this parameter should be equal to or lower than the time required by the LiDAR to generate a new scan to guarantee that all watermarked scans contain different concealed information, lowering the chances of replay attacks.

C. Watermarking process

Our goal is to propose a watermark process that is: *Fragile*, since the watermark is susceptible to unwanted modifications so that tampering can be detected. *Invisible*, since it can apply a



(a) Representation of the point cloud as seen through the watermarking



(b) Division of the space in regions over the Y-axis, T is the thickness of each region. The blue rectangle represents the region mean (M) in relation to the Y-axis.

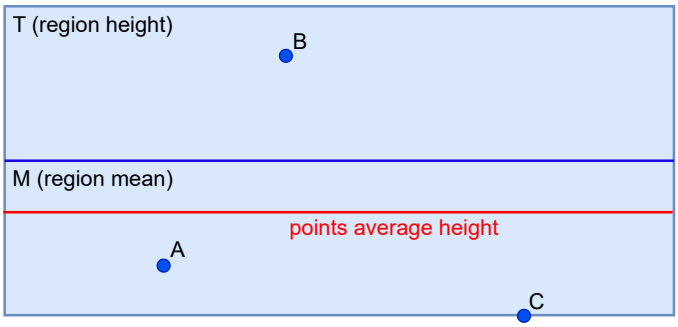
Fig. 3: Division of the space in the watermark process.

small enough displacement to points that it becomes challenging to distinguish between the original and the watermarked point clouds. *Imperceptible*, as ADAS inference operations on the watermarked point cloud produce results that are almost identical to those obtained with the original point cloud, and finally *Blind*, since the receiver does not need to acquire the original point cloud to extract the watermark.

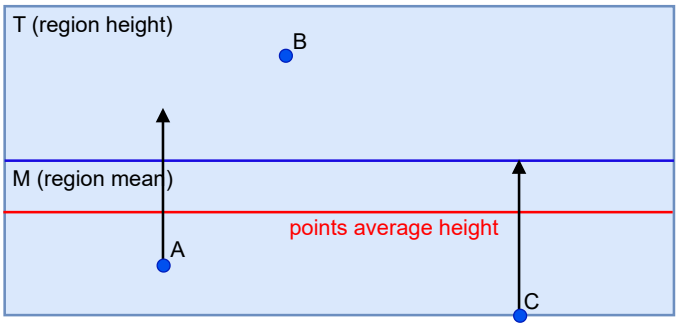
Similarly to previous works [3], [14], the proposed fragile watermarking approach involves dividing the point cloud's 3D space into several regions, where each region represents a hidden information bit. The value of these bits is regulated during the watermark embedding phase.

Our watermarking process comprises three distinct watermarking operations that are independently applied to each of the three axes (X, Y, Z). For each axis, we approximate the point cloud space as a parallelepiped, allowing the space to be partitioned into identical regions. The space is divided into small parallelepipeds with uniform thickness (T), as represented in Figure 3.

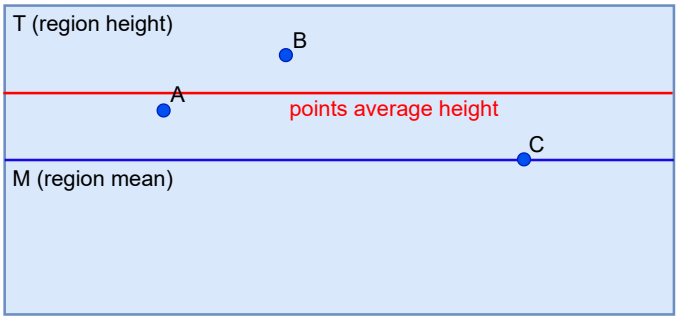
The selection of an appropriate value T defining the height of each region for each of the axes is a crucial factor in determining the optimal settings for the watermarking algorithm. A tradeoff between three elements, the imperceptibility of the watermark, the effectiveness of the tampering detection process, and the computational efficiency, is necessary. In fact, a higher value of T would lead to fewer regions, thereby reducing the time and resource requirements of the algorithm. However, this would come at the cost of a deterioration in



(a) Region before watermark embedding phase.



(b) Watermark embedding process.



(c) Region after watermark embedding phase.

Fig. 4: Division of the space in the watermark process.

tampering detection performance and in watermark imperceptibility. Conversely, a lower T value results in smaller point displacements to correct the average height value of each region, providing greater precision in detecting the tampering attacks and increasing the watermark imperceptibility.

Watermark embedding. During the watermark embedding phase, for all regions defined before, a bit is extracted from the sequence of generated bits that compose the hidden message, and a corresponding region is selected from the point cloud space. The selection process starts with regions in the X coordinate and then moves on to those in the Y and Z coordinates.

To embed information in each region, we calculate the average height of the points inside the region and compare it with half the height of the region itself (region mean "M"). If the calculated average is greater or equal to M, the embedded information is a binary 1; otherwise, it is a binary 0. If the information does not correspond to the desired value, it is corrected by applying a positive or negative displacement of

value M to each point in the subregion where the current average lies. Note that by displacing the points of a positive or negative M, we ensure that the average point height becomes equal to or higher than M itself in any configuration of points inside the region while never exceeding the boundaries of the region. Figure 4 illustrates an example of this operation. Repeating the process for all regions in the three axes, each point is considered part of three regions, one per axis. Therefore, modifying a single point affects three bits of information.

Figure 4a provides a 2D representation of a region containing three points. The average value of the points in the region M is depicted by the black line, while the current calculated average is illustrated by the red one. The objective is to embed a bit with a value of 1 in that region; thus, the current average value needs to be corrected. Figure 4b shows the process of embedding, which involves shifting the positions of points "A" and "B" upwards of a value M. The results of the embedding process are depicted in Figure 4c, where the new current average position is coherent with the bit information that needs to be embedded.

There is a possibility for a region to contain no points. The probability of this occurrence increases as the T parameter decreases. If an empty region is encountered during a round, the algorithm moves to the next region but does not move to the next bit in the bit sequence. The reason behind this choice (instead of ignoring the bit in the empty region) derives from the necessity of avoiding object removal attacks. In fact, if the empty region does not affect the rest of the process, the attacker may remove points without being detected. Moreover, by inserting the hidden bit in the next non-empty region, any manipulation that involves an empty region can potentially break the entire watermark. If the attacker inserts points in the empty region, it anticipates, in the extraction process, the use of the next bit of the hidden message, creating a shift in the bit sequence reading order. The attacker is, therefore, forbidden from emptying a region containing points, inserting points in an empty region, or changing region order in the point cloud.

Watermark extraction. Similarly to the watermark injection phase, the watermark extraction phase is carried out in rounds. In each round, if the region is not empty, the receiver computes the hidden bit of the region and compares it with the current bit of the known hidden message. If the two values correspond, then the region is considered not tampered with, and the process proceeds to the subsequent round. If the values do not correspond, the region is flagged as tampered, and the process proceeds to the subsequent round. Note that even in the case where the region is considered tampered, it is possible to detect which areas of the point cloud were subjects of the attack by completing the evaluation of all regions.

IV. Evaluation

Our evaluation aims to demonstrate our system's real-world effectiveness, ensuring that it does not affect the normal functioning of the LiDAR object detection process and demonstrating its attack detection capabilities. To ensure that it does not introduce excessive degradation on the results of inference operations on point cloud data, we test the imperceptibility of the watermarked point cloud by evaluating the shift in the bounding boxes generated by the object detector. Moreover, we test the timing overhead generated by the algorithm to ensure it does not exceed the requirements. To evaluate the attack detection capabilities of our system, we test it against object insertion and object removal attacks, as discussed in Section III-A, and evaluate its performance in detecting them.

TABLE I: Hardware specifications

	Specifications
CPU	Intel(R) Core(TM) i7-1065G7 CPU @ 1.30GHz
GPU	Intel(R) Iris(R) Plus Graphics
RAM	16 GB

We report the results of two different T parameter values that, upon empirical analysis, we deem valid for the automotive application. The first has a flat T value of 10 cm, while the second varies depending on the distance from the LiDAR sensor. Specifically, regions between -20m and +20m were assigned a T value of 10cm, while regions in the remaining space were assigned a doubled T value of 20cm. As a result, displacement values for points in larger regions were doubled, becoming 10cm or 4cm depending on their proximity to the average.

A. Evaluation Setup

We use data obtained from the KITTI dataset [11] to validate the implementation of our proposed solution. The KITTI dataset is widely recognized in the autonomous driving domain and contains various types of data obtained from different sensors, including a LiDAR Velodyne HDL-64E rotating 3D laser scanner.

Each Velodyne scan in the KITTI dataset is a binary file that contains a list of points collected in that particular frame. Each point is represented with four floating point values, where the first three are used to locate the point in space and represent the respective values of the point in each coordinate (X, Y, Z). The last value is the reflectance value, which represents the power of the received reflected laser beam for that point, but this is not useful for our purposes. The number of points in each frame varies, but the authors estimate an average of 120,000 points per frame.

To properly implement and test our solution, we determine the maximum and minimum values that a point could assume in each coordinate. The maximum and minimum values for X and Y were around +80 meters and -80 meters, respectively. The maximum and minimum values for Z were around +33 meters and -2 meters, respectively.

We execute our tests on a machine with Intel(R) Core(TM) i7-1065G7 CPU, Intel Iris graphics, and 16 GB memory, and on Google Colab (timing analysis was performed fully on our Windows machine. Given that currently marketed vehicles mount Intel Core, NXP i.MX CPUs and NVIDIA Drive, AMD Radeon GPUs for infotainment, and ADAS systems, we find our hardware comparable with automotive ones). The watermark embedding and extraction code and the experiments were compiled and executed via the CLion interface. To evaluate the effectiveness of our approach, we use the PointPillars [12] object detector, a well-established 3D object detection algorithm frequently utilized in autonomous driving and robotics applications. The OpenPCDet framework,

an open-source framework based on PyTorch for 3D point cloud object detection tasks, used for the implementation of a variety of state-of-the-art detection algorithms [26], was employed to implement the object detector.

B. Imperceptibility evaluation

The purpose of the imperceptibility evaluation is to ensure that the results of the operations performed on the original and tampered scans are nearly indistinguishable. To conduct this test, we employ the object detector to conduct inferences on both scans and compare the outcomes. We test the results on the bounding boxes and not on the point cloud directly to ensure that the modifications generated by our system do not limit the usability of the data. We calculate the difference between each value of the bounding box in the original scan and its corresponding value in the marked scan to carry out the comparison. We conduct the experiment on 16 distinct scans of the test set, and the worst results obtained for each bounding box variable comparison are presented in Table II.

The obtained results indicate that the bounding boxes extracted from the tampered scan exhibit only slight differences with respect to the original bounding boxes. The vertices of the bounding boxes may be shifted by less than 10cm in each coordinate, while the length of bounding box sides may vary by a maximum of approximately 10cm. The largest shift observed for the rotational value of a bounding box is approximately 3 degrees. For the automotive application, the acceptable angular resolution of short- and long-range LiDAR is respectively 1° at 30 meters and 0.15° at 300 meters [28], [7], obtaining an error of approximately half a meter, which is significantly higher than the differences generated by our approach. Nonetheless, if necessary, a modification of the T parameter could lower such differences at the cost of time or computation requirements. For example, for applications that have fewer computation constraints but require higher precision, such as mapping for virtual reality, the value of T could be increased to lower the differences.

C. Timing evaluation

A timing evaluation is a fundamental procedure aimed at measuring the computational efficiency of the watermark embedding and extraction algorithms in a soft real-time context. We execute each algorithm multiple times, specifically forty, and calculate the time required for each operation. We compute both the average and the worst-case scenario; the results are presented in Table III.

The results indicate that the watermark embedding and extraction algorithms are able to meet the required time constraints, as the LiDAR sensor rotates at a frequency of 10Hz [11], and thus the watermarking operations must be performed in less than 0.1 seconds. A modification in the value of the T parameter, which defines the thickness of each region, would impact the timing requirements of the algorithm. Depending on the application, the hardware capabilities of the sender and receiver, and the real-time requirements of the context at hand, it is possible to choose the most fitting T value.

D. Object insertion attack evaluation

To evaluate the effectiveness of the proposed approach in defending against insertion attacks, we develop an implementation of the attack described in Section III-A. The attack involves selecting an object in the marked point cloud and creating a copy of it to be inserted into the same scan at a different position. This technique was chosen because it is a reliable form of tampering in which the object detection algorithm correctly identifies the newly inserted objects. To perform the attack, the attacker inputs the marked point cloud into the object detection algorithm used by the ADAS system. Then, the attacker chooses one of the detected objects and retrieves the respective bounding box values. The attack takes the bounding box values and marked point cloud as input, along with three numerical values indicating shifts in the position applied to the three different coordinates of points in the point cloud for the selected object (X, Y, Z). It utilizes the bounding box information to select points representing the respective object in space and inserts new points into the scan that are identical to those points but shifted by the three numerical values inserted at the start. The output of the process is the newly tampered scan.

To test the correctness of the proposed approach, we utilize the tampered scans as input to the watermark extraction process and consider the tampering as detected when one or more regions in space corresponding to the inserted object coordinates are marked as tampered or when, due to an empty space being filled during tampering, an entire section of the point cloud is considered tampered.

We conduct the experiment on 14 different scans from the test set, selecting all visible objects for each scan and duplicating the object in a thousand different positions within the scan for a thousand output point clouds. The final results, as shown in Table IV, demonstrate that the proposed approach is successful in up to **98%** of cases. It is important to note that these results are dependent on various factors, particularly the T value, object detector, and LiDAR sensor used to capture the scan.

TABLE IV: Tampering detection results comparison.

	accuracy
insertion attack (T = 10)	0.986
removal attack (T = 10)	0.941
insertion attack (T varies)	0.983
removal attack (T varies)	0.930

E. Object removal attack evaluation

The purpose of the object removal test is to evaluate the effectiveness of the proposed approach in defending against removal attacks as described in III-A. To conduct this test, we follow similar steps to those taken for the object insertion attack. We input the marked scan in the object detector, selected one of the detected objects, and collected its corresponding bounding box values. Subsequently, we identify the points in the point cloud that correspond to the selected object (i.e., the points within the respective bounding box) and remove them from the scan. The resulting output is a new point cloud from which the points corresponding to the selected object have been removed.

TABLE II: Imperceptibility test results comparison.

	X	Y	Z	L	W	H	R
max difference (T = 10)	0.0872 m	0.0593 m	0.0592 m	0.1154 m	0.0716 m	0.0708 m	0.0528 rad
max difference (T varies)	0.1723 m	0.0951 m	0.0702 m	0.2641 m	0.0724 m	0.0871 m	0.0719 rad

TABLE III: Timing test results comparison.

	maximum	average
embedding (T = 10)	0.05637 s	0.039231 s
extraction (T = 10)	0.03803 s	0.031571 s
embedding (T varies)	0.04959 s	0.041437 s
extraction (T varies)	0.03745 s	0.031418 s

To test the accuracy of our approach, we employ the watermark extractor to determine whether at least one region of space corresponding to the object's position was marked as tampered. We perform the object removal test on 25 different scans from the test set, utilizing all the objects detected in each scan. There were 68 objects in total. The algorithm detected 64 object removal attacks, resulting in an accuracy of **94%**. It is important to note that the accuracy of the approach may vary considerably based on the T value, object detector, and LiDAR sensor used to capture the scan.

V. Related works

The problem of detecting tampering with LiDAR scans in AVs has already been studied in the literature. The choice of using watermarking instead of other techniques (e.g., encryption, HMACs) to ensure the integrity of LiDAR point cloud data stems from the low computational costs of watermarking and from the capability of watermarking algorithms to recognize the region under attack and not consider the point cloud data as a single entity while being considered a strong security mechanism.

The solutions proposed by Bahirat et al. [1] and by Ponto et al. [21] involve checking for inconsistencies in the point cloud's characteristics to detect tampering attempts. Indeed, an attacker attempting to manipulate the point cloud could implement changes that are unfeasible, such as inserting points in a position that is unreachable by sensor beams or inserting point clouds of new elements with different densities. However, these approaches guarantee no protection against object removal and replay attacks and rely on the assumption that the attacker cannot tamper with the data in a way that is coherent with the characteristics of an authentic scan.

An alternative solution, proposed by Liu et al. [13], is to compare the data obtained from LiDAR point clouds with data obtained from other sensors, such as cameras and radar. If the object detection results of LiDAR data differ significantly from those of the other sensors, then the point cloud is considered to have been tampered with. However, this approach has limitations, such as the difficulty of identifying which sensor has been compromised in the event of detecting any inconsistency. Additionally, the concept of using a broad range of data to perform an integrity check is inherently flawed, as these data must be used in conjunction to generate a more precise perception layer, and redundant information could be essential in case any sensor is unavailable due to a malfunction.

There are also approaches based on fragile digital watermarking techniques. ALERT (Authentication, Localization, and Estimation of Risks and Threats), proposed by Bahirat et al. [2], uses a dynamic watermark that changes in every frame and is derived from the depth map obtained from stereo data. The depth map information is embedded in the point cloud watermark, while the metadata needed to extract the point cloud watermark is hidden in the stereo data watermark. In ADAS, the first step is to extract the stereo data watermark and derive the metadata and depth map information. Then, the metadata are used to extract the hidden point cloud information, which is subsequently compared with the generated depth map to verify correctness. The dynamic nature of the watermark enhances the security of LiDAR data. However, there are several drawbacks to this approach. Firstly, the security of LiDAR data is dependent on the security of camera data. Therefore, a single attack against camera data could compromise the security of both sensors. Secondly, the approach is inefficient in terms of computation and timing, as the point cloud watermark extraction requires the extraction of the camera watermark first, rendering the approach unfeasible for real-time applications. Lastly, the approach does not offer protection against replay attacks, as an attacker could simply resend the same corresponding camera and point cloud frames to break security.

An alternative approach, described by Changalvala et al. [3], proposes a technique based on 3D Quantization Index Modulation (QIM) data hiding [5], [6]. The method involves dividing the point cloud space into fixed-size voxels and quantizing the points inside each voxel to a single point that is one of its vertices. All points in the point cloud that fall in a given voxel are approximated to a single point, positioned in a specific location inside the region, depending on the hidden value to hide. Specifically, each voxel is imagined to contain a cube with eight vertexes. During the watermark embedding phase, the positions of the vertices are then mapped to predefined positions inside the respective voxel to hide information. Depending on the vertex in which the approximated point is positioned, it is possible to embed 3-bit long information in each voxel. On the receiver side, the same quantization step is performed, and the hidden information is extracted by analyzing the displacement of vertices with respect to the original position. This information is then compared with the known one to verify the integrity of the scan and locate any tampered region.

A final solution, described by Long et al. [14], relies on two separate operations: watermark verification and similarity detection. The authors execute similarity detection between two consecutive LiDAR scans to detect the non-overlapping areas of the new scan. The newly arrived LiDAR scan is compared with the previous one using Hausdorff distance [23]. Only regions that have changed more than a certain threshold are chosen to embed the watermark.

The watermarking process involves dividing the point cloud space into concentric adjacent spherical rings, with the LiDAR sensor at the center. Each spherical ring corresponds to a single bit of hidden information obtained by analyzing the

position of the average radius of points inside it. For each region, the points chosen in the previous step that fall in such region are moved towards or away from the center point of the spherical ring to embed the watermark. The watermark embedding process is based on adjusting the average radius of points within each ring by applying a shift in points' position to embed a hidden bit each. In the extraction phase, the same space division is applied, and by inspecting the value of the average radius of points in each ring, it is possible to extract the embedded bit of information.

A. State-of-the-art comparison

While we report a tabular comparison with the aforementioned literature in Table V, we focus our comparison on the two works in the state of the art, to the best of the authors' knowledge, that approach LiDAR data tampering detection through watermarking without requiring additional data sources: Changalvala et al.'s 3D data hiding technique [3] and Long et al.'s spherical rings technique [14].

On computation and timing. The authors of [3] do not execute a timing analysis, while the authors of [14] report an average of 0.01 s for embedding and 0.096 s for extraction and detection, similar to our results. More interestingly, changing the dimension of the region under analysis (in order to increase the accuracy and imperceptibility of the algorithm) leads to different computation costs: assuming that the computation cost for a single region is approximately similar, in our approach, changing the dimension of the regions linearly increases the number of regions under consideration. While [14] behaves similarly, [3] has a cubic increase in voxels in relation to their side dimensions, as shown in Table V.

On region tampering detection capabilities. The authors of [3] do not explicitly report detection rates, while the authors of [14] report an accuracy of 0.93 to 0.98 and a false positive rate of 0.08 to 0.31, depending on the driving conditions. Our approach has similar accuracy but has generated 0 false positives throughout our evaluation. It is relevant to understand, on top of the detection rates, the capabilities of each approach to detect the areas that have been tampered with. In the case of [3], given that each voxel is evaluated by itself, in the ideal conditions, the algorithm should be capable of delimiting the region under attack with an error of under a voxel. In the case of [14], however, the detection process is executed on each spherical ring; hence, in the best-case scenario, the defense mechanism is capable of defining a range of distances from the LiDAR sensor at which the attack was implemented. In our approach, a modified point affects three regions, one per axis. In the best-case scenario, where all three regions change the watermark value, the tampered area is delimited by a cube of height T. In case only one of the regions changes value, the detected area is a cuboid. It is important to note that an attack usually modifies multiple points affecting many regions, increasing the ease of delimiting an area of attack.

On the effectiveness against strong attackers. A watermarking technique for tampering detection must be effective even against strong attackers, where for *strong*, we define an attacker that has knowledge of the defense system and its functioning (while not knowing the hidden message to embed) and significant computational capabilities. We have found two vulnerabilities in the techniques presented in the state-of-the-art that make them avoidable by a strong attacker.

In the case of [3], the vulnerability lies in the lack of handling of empty voxels. While the authors do not explicitly mention how they handle empty voxels, two approaches are available, as we mentioned in Section III-C: if the algorithm, upon encountering an empty voxel, ignores it and skips the related hidden information, then the attacker can easily implement object removal attacks. If the algorithm, instead, does not skip the related information, it is trivial for the attacker to recover it from the first successive non-empty voxel and repeat this process for the whole injection to empty a number of voxels equal to those created to balance the total hidden information in the point cloud. This vulnerability highlights one of the strengths of our approach. Since each point is related to three different regions, each with its own hidden information, although the attacker may recover the information embedded in a specific region, he cannot modify it without the risk of modifying the rest of the regions affected by the change.

In the case of [14], the vulnerability lies in the single dimension of the detection process. In fact, since the watermark is embedded in the distance of a set of points from the center point of the spherical ring, the attacker can theoretically move the points where he prefers as long as he maintains the distance of each point from the center of the ring; for example moving them from in front to behind the vehicle, to remove an obstacle, or vice-versa. Again, this vulnerability highlights the strength of our approach of using multiple dimensions, which do not enable the attacker to hide in the dimensions that are not evaluated by the detection approach.

VI. Conclusions

In this paper, we proposed a novel solution to the problem of LiDAR data tampering. Our solution involves embedding a fragile watermark in the LiDAR point cloud to recognize the areas of the point cloud that have been tampered with. We validated our approach, demonstrating its feasibility in real-world scenarios by analyzing its timing requirements and imperceptibility. We evaluated its effectiveness in detecting injection and removal attacks, obtaining promising results. Finally, we compared our work with current state-of-the-art LiDAR watermarking approaches, discussing strengths and shortcomings. Future works will focus on better evaluating the detection capabilities of our approach, specifically focusing on delimiting the tampered areas in sub-optimal scenarios.

Acknowledgment

This study was carried out within the MICS (Made in Italy – Circular and Sustainable) Extended Partnership and received funding from Next-Generation EU (Italian PNRR – M4 C2, Invest 1.3 – D.D. 1551.11-10-2022, PE00000004). CUP MICS D43C22003120001

References

[1] K. Bahirat and B. Prabhakaran, "A study on lidar data forensics," in *2017 IEEE International Conference on Multimedia and Expo, ICME 2017, Hong Kong, China, July 10-14, 2017*. IEEE Computer Society, 2017, pp. 679–684. [Online]. Available: https://doi.org/10.1109/ICME.2017.8019395

TABLE V: Comparison with techniques in literature for LiDAR point cloud watermarking and inconsistency detection. An empty cell means that the column is not applicable to the given approach. Note that computation times are only intended as a reference to the real-time claim and not as a comparison since datasets, approaches, and hardware may differ between the various solutions.

	Method	Real-time	Computation time	D/C^2	Empty Space	Multi-dimension
Bahirat et al. [1]	Point-cloud consistency	No	N/A	N/A		
Ponto et al. [21]	Point-cloud consistency	No	N/A	N/A		
LIFE [13]	Coherency with camera	High-end GPUs	0.108s	N/A		
ALERT [2]	Quantization + stereo-camera	Yes	>0.3s	N/A	✓	✓
3D data hiding [3]	Voxel-based watermark	Yes	N/A	n^3		✓
Spherical rings [14]	Distance-based watermark	Yes	0.088s	n	✓	
Our approach	3D watermark embedding	Yes	0.056s	$3n$	✓	✓

[2] K. Bahirat, U. Shah, A. A. Cárdenas, and B. Prabhakaran, "ALERT: adding a secure layer in decision support for advanced driver assistance system (ADAS)," in *2018 ACM Multimedia Conference, MM 2018, Seoul, Republic of Korea, October 22-26, 2018*, S. Boll, K. M. Lee, J. Luo, W. Zhu, H. Byun, C. W. Chen, R. Lienhart, and T. Mei, Eds. ACM, 2018, pp. 1984–1992. [Online]. Available: https://doi.org/10.1145/3240508.3241912

[3] R. Changalvala and H. Malik, "Lidar data integrity verification for autonomous vehicle using 3d data hiding," in *IEEE Symposium Series on Computational Intelligence, SSCI 2019, Xiamen, China, December 6-9, 2019*. IEEE, 2019, pp. 1219–1225. [Online]. Available: https://doi.org/10.1109/SSCI44817.2019.9002737

[4] S. Checkoway, D. McCoy, B. Kantor, D. Anderson, H. Shacham, S. Savage, K. Koscher, A. Czeskis, F. Roesner, and T. Kohno, "Comprehensive experimental analyses of automotive attack surfaces," in *20th USENIX Security Symposium, San Francisco, CA, USA, August 8-12, 2011, Proceedings*. USENIX Association, 2011. [Online]. Available: http://static.usenix.org/events/sec11/tech/full_papers/Checkoway.pdf

[5] B. Chen and G. W. Wornell, "Digital watermarking and information embedding using dither modulation," in *Second IEEE Workshop on Multimedia Signal Processing, MMSP 1998, Redondo Beach, California, USA, December 7-9, 1998*, P. W. Wong, A. Alwan, A. Ortega, C. J. Kuo, and C. L. M. Nikias, Eds. IEEE, 1998, pp. 273–278. [Online]. Available: https://doi.org/10.1109/MMSP.1998.738946

[6] ——, "Quantization index modulation: A class of provably good methods for digital watermarking and information embedding," *IEEE Trans. Inf. Theory*, vol. 47, no. 4, pp. 1423–1443, 2001. [Online]. Available: https://doi.org/10.1109/18.923725

[7] Z. Dai, A. Wolf, P. Ley, T. Glück, M. C. Sundermeier, and R. Lachmayer, "Requirements for automotive lidar systems," *Sensors*, vol. 22, no. 19, p. 7532, 2022. [Online]. Available: https://doi.org/10.3390/s22197532

[8] A. de Faveri Tron, S. Longari, M. Carminati, M. Polino, and S. Zanero, "Canflict: Exploiting peripheral conflicts for data-link layer attacks on automotive networks," in *Proceedings of the 2022 ACM SIGSAC Conference on Computer and Communications Security, CCS 2022, Los Angeles, CA, USA, November 7-11, 2022*, H. Yin, A. Stavrou, C. Cremers, and E. Shi, Eds. ACM, 2022, pp. 711–723. [Online]. Available: https://doi.org/10.1145/3548606.3560618

[9] A. A. Embaby, M. A. W. Shalaby, and K. M. Elsayed, "Digital watermarking properties, classification and techniques," *International Journal of Engineering and Advanced Technology (IJEAT)*, vol. 9, no. 3, pp. 2742–2750, 2020.

[10] D. J. Fagnant and K. Kockelman, "Preparing a nation for autonomous vehicles: opportunities, barriers and policy recommendations," *Transportation Research Part A: Policy and Practice*, vol. 77, pp. 167–181, 2015.

[11] A. Geiger, P. Lenz, C. Stiller, and R. Urtasun, "Vision meets robotics: The KITTI dataset," *Int. J. Robotics Res.*, vol. 32, no. 11, pp. 1231–1237, 2013. [Online]. Available: https://doi.org/10.1177/0278364913491297

[12] A. H. Lang, S. Vora, H. Caesar, L. Zhou, J. Yang, and O. Beijbom, "Pointpillars: Fast encoders for object detection from point clouds," pp. 12 697–12 705, 2019.

[13] J. Liu and J. Park, ""seeing is not always believing": Detecting perception error attacks against autonomous vehicles," *IEEE Trans. Dependable Secur. Comput.*, vol. 18, no. 5, pp. 2209–2223, 2021. [Online]. Available: https://doi.org/10.1109/TDSC.2021.3078111

[14] T. Long, A. Xie, X. Ren, and X. Wang, "Tampering detection of lidar data for autonomous vehicles," in *2021 40th Chinese Control Conference (CCC)*. IEEE, 2021, pp. 4732–4737.

[15] C. Miller and C. Valasek, "Adventures in automotive networks and control units," *Def Con*, vol. 21, no. 260-264, pp. 15–31, 2013.

[16] ——, "Remote exploitation of an unaltered passenger vehicle," *Black Hat USA*, vol. 2015, no. S 91, pp. 1–91, 2015.

[17] ——, "Advanced can injection techniques for vehicle networks," *Black Hat USA, Las Vegas, NV, USA*, vol. 4, 2016.

[18] D. M'Raihi, M. Bellare, F. Hoornaert, D. Naccache, and O. Ranen, "Hotp: An hmac-based one-time password algorithm," Tech. Rep., 2005.

[19] D. M'Raïhi, S. Machani, M. Pei, and J. Rydell, "TOTP: time-based one-time password algorithm," Tech. Rep., 2011. [Online]. Available: https://doi.org/10.17487/RFC6238

[20] H. Nyeem, W. W. Boles, and C. Boyd, "Digital image watermarking: its formal model, fundamental properties and possible attacks," *EURASIP J. Adv. Signal Process.*, vol. 2014, p. 135, 2014. [Online]. Available: https://doi.org/10.1186/1687-6180-2014-135

[21] K. Ponto, S. Smith, and R. Tredinnick, "Methods for detecting manipulations in 3d scan data," *Digit. Investig.*, vol. 30, pp. 101–107, 2019. [Online]. Available: https://doi.org/10.1016/j.diin.2019.07.009

[22] C. Qin, P. Ji, X. Zhang, J. Dong, and J. Wang, "Fragile image watermarking with pixel-wise recovery based on overlapping embedding strategy," *Signal Process.*, vol. 138, pp. 280–293, 2017. [Online]. Available: https://doi.org/10.1016/j.sigpro.2017.03.033

[23] G. Rote, "Computing the minimum hausdorff distance between two point sets on a line under translation," *Inf. Process. Lett.*, vol. 38, no. 3, pp. 123–127, 1991. [Online]. Available: https://doi.org/10.1016/0020-0190(91)90233-8

[24] A. Shaik, V. Thanikaise, and R. Amirtharajan, "Data security through data hiding in images: A review," *Journal of Artificial Intelligence*, vol. 10, pp. 1–21, 12 2016.

[25] L. Singh, A. K. Singh, and P. K. Singh, "Secure data hiding techniques: a survey," *Multim. Tools Appl.*, vol. 79, no. 23-24, pp. 15 901–15 921, 2020. [Online]. Available: https://doi.org/10.1007/s11042-018-6407-5

[26] O. D. Team, "Openpcdet: An open-source toolbox for 3d object detection from point clouds," https://github.com/open-mmlab/OpenPCDet, 2020.

[27] J. Z. Varghese, R. G. Boone *et al.*, "Overview of autonomous vehicle sensors and systems," in *International Conference on Operations Excellence and Service Engineering*. sn, 2015, pp. 178–191.

[28] M. E. Warren, "Automotive LIDAR technology," in *2019 Symposium on VLSI Circuits, Kyoto, Japan, June 9-14, 2019*. IEEE, 2019, p. 254. [Online]. Available: https://doi.org/10.23919/VLSIC.2019.8777993