Sensor Data Integrity Verification for Real-time and Resource Constrained Systems

by

Raghavendar Changalvala

A dissertation submitted in partial fulfillment
of the requirements for the degree of
Doctor of Philosophy
(Electrical and Computer Engineering)
in the University of Michigan-Dearborn
2021

Doctoral Committee:

Professor Hafiz Malik, Chair
Assistant Professor Anys Bacha
Ashok Prajapati, General Dynamics Land Systems
Associate Professor Samir Rawashdeh
Professor Weidong Xiang

Raghavendar Changalvala

rchangal@umich.edu

ORCID iD: 0000-0001-5665-2401

DEDICATION

This dissertation is dedicated to my parents, family, and all the mentors that served as my source of wisdom.

ACKNOWLEDGEMENTS

First of all, I would like to express my sincere gratitude towards my advisor, Dr. Hafiz Malik for his continuous support to my research and his endless motivation. The meetings and conversations I had with you were vital in inspiring me to think outside the box and from multiple perspectives. Further, I would like to thank each one of my dissertation committee members for their thoughtful comments and recommendations on this dissertation. To conclude, I cannot forget to thank my family, friends and work life managers and colleagues for all the unconditional support they provided in this very intense academic journey.

TABLE OF CONTENTS

DEDICATION ii

ACKNOWLEDGEMENTS iii

LIST OF FIGURES viii

LIST OF TABLES xii

LIST OF APPENDICES xiii

LIST OF ABBREVIATIONS xiv

ABSTRACT xvi

CHAPTER

I. Introduction 1

1.1 Introduction 1
1.2 Attacks on Sensors 4
1.2.1 Automotive Sensors Vulnerability 5
1.2.2 Automotive Sensor Categories 7
1.3 Problem Statement 9
1.4 Watermarking Advantages 13
1.5 Thesis Outline 16
1.5.1 Document Road Map 17

II. Data Security Using Digital Watermarking 19

2.1 Watermarking Background 19
2.2 Applications of Watermarking 20
2.2.1 Content Identification and Protection 22
2.2.2 Digital Forensics and Piracy Detection 22
2.2.3 Ownership and Copyright Protection 22
2.2.4 Content Security and Authentication 23

2.2.5 Location of Content Online . . . 23
2.2.6 Broadcast Monitoring . . . 23
2.2.7 Auditing . . . 24
2.2.8 Access Control . . . 24
2.2.9 Medical Applications . . . 24
2.2.10 Clandestine Communication or Steganography . . . 24
2.3 Watermark Security . . . 25
2.3.1 Threat Models . . . 26
2.3.1.1 Removal Attack . . . 27
2.3.1.2 Interference Attack . . . 27
2.3.1.3 Geometric Attack . . . 27
2.3.1.4 Filtering Attack . . . 27
2.3.1.5 Active Attack . . . 27
2.3.1.6 Passive Attacks . . . 28
2.3.1.7 Data Degradation . . . 28
2.3.2 Watermarking & Cryptography Analogy . . . 29
2.4 Watermark Design Requirements . . . 31
2.4.1 Imperceptibility . . . 32
2.4.1.1 Hausdorff Distance . . . 32
2.4.1.2 Signal-to-Noise Ratio (SNR) . . . 33
2.4.1.3 Root Mean Square Error (RMSE) . . . 33
2.4.2 Tamper Resistance . . . 34
2.4.3 Robustness . . . 34
2.4.4 Security . . . 35
2.4.5 Capacity . . . 36
2.4.6 Computational Cost . . . 36
2.4.7 False Positive & False Negative . . . 37
2.4.8 Watermark Keys . . . 38
2.4.9 Watermark Reversibility . . . 38
2.5 Watermarking Techniques . . . 38
2.5.0.1 Spatial Domain Watermarking . . . 39
2.5.0.2 LSB . . . 41
2.5.0.3 1D QIM . . . 42
2.5.1 Frequency Domain Watermarking . . . 43
2.5.2 Perception Based Watermarking . . . 43
2.5.2.1 Fragile Watermarking . . . 44
2.5.2.2 Semi-fragile Watermarking . . . 44
2.5.2.3 Robust Watermarking . . . 45

III. Watermarking & Data Models . . . 46

3.1 Watermarking Model . . . 46
3.1.1 Data Model . . . 46
3.2 Geometric and Statistical Models . . . 49
3.3 Communication Model . . . 53

3.4 Scalar Watermarking 59
3.4.1 Binning Schemes - LSB 60
3.4.2 Quantization Index Modulation - QIM 61
3.5 Lattice Codes 65
3.5.1 Lattice QIM 67

IV. RADAR Data Integrity Verification-2D QIM 72

4.1 Introduction 73
4.2 System & Attack Model 76
4.2.1 Sensor Fusion Data Model 77
4.3 Proposed Framework 79
4.3.1 Watermark Generation 81
4.3.2 Watermark Embedding 82
4.3.3 Watermark Decoding 84
4.4 Security Analysis & Performance Evaluation 85
4.4.1 Data Addition 86
4.4.2 Data Deletion 87
4.4.3 Data Modification 88
4.5 Experiments & Results 89
4.5.1 Impact of Embedding Distortion on Object Detection 89
4.5.2 Bit Error Rate 98
4.5.3 False-alarm Rate Analysis 99

V. LiDAR Data Integrity Verification-3D QIM 102

5.1 Introduction 103
5.2 LiDAR Point Cloud: Applications 104
5.2.1 QIM-based Data Hiding on LiDAR Point Cloud . . 105
5.3 Attack Modeling 108
5.3.1 Attack Vectors 110
5.4 Countermeasures to Transmission Channel Attacks 111
5.4.1 Implementation Details 111
5.4.2 Performance Evaluation 112
5.5 Experimental Results 113
5.5.1 Impact of Embedding Distortion on ADAS Performance 115
5.5.2 Embedding Distortion Analysis 119
5.5.3 Robustness Analysis 120
5.5.3.1 Bit Error Rate 123
5.5.3.2 Tamper Detection and Localization . . . 124
5.5.3.3 False-alarm Rate Analysis 127
5.6 Vulnerability Analysis of QIM 130
5.6.1 Countermeasure Framework 132
5.6.1.1 Dither Modulation 132

5.6.1.2 Watermark Generation 134
5.6.1.3 Watermark Embedding 135
5.6.2 Experiments & Results 136
5.6.2.1 Dataset 136
5.6.2.2 Bit Error Rate 137
5.6.2.3 Localization Accuracy 137
5.6.2.4 False Negatives 139

VI. Future Work & Conclusion 143

6.1 Need for Data Security in Autonomous Vehicles 143
6.1.1 Data Sources 145
6.1.2 Framework Proposal 146
6.2 Sensor Fingerprints 151
6.2.1 Methodology 153
6.2.2 System Model 153
6.2.3 Data Model 155
6.2.4 Threat Model 156
6.2.5 Fingerprint Extraction 157
6.2.6 Experiments & Results 160
6.3 Conclusion 162

APPENDICES 165

BIBLIOGRAPHY 170

LIST OF FIGURES

FIGURE

1.1	Autonomous vehicle sensor suite (Snehaprabha and Ram, 2019)	3
1.2	Autonomous vehicle sensor suite - Raw sensor	8
1.3	Autonomous vehicle sensor suite - Smart sensor	9
1.4	Comparison of different methods to achieve sensor data integrity	12
2.1	Watermarking requirements	31
2.2	Watermarking techniques classification	39
2.3	Amplitude plots of a raw and LSB modified audio sample	40
2.4	Representation of raw and LSB modified camera image	41
2.5	Representation of raw and QIM modified LiDAR image	42
3.1	Digital content representation	49
3.2	Geometrical model representation	51
3.3	Watermarking as communication model	54
3.4	1D QIM scheme	62
3.5	Simple 1D and 2D lattices	66
3.6	Quincunx lattice	67
3.7	Hexagonal lattice QIM scheme	69

4.1 Radar data stream 74

4.2 Block-diagram of problem statement 75

4.3 State vector for pedestrian motion 77

4.4 Proposed framework and 2D QIM embedding process 78

4.5 Block-diagram of proposed method 79

4.6 Time-stamp conversion 80

4.7 2D QIM scheme 82

4.8 Data addition attack vector depiction 85

4.9 Data deletion attack vector depiction 88

4.10 Data modification attack vector depiction 89

4.11 Tamper localization algorithm performance under varying channel noise 91

4.12 RMSE comparison at $R_m = 0.4$ 92

4.13 RMSE comparison at $R_m = 0.5$ 93

4.14 Comparison: EKF path prediction from clean and encoded data at $R_m = 0.5$, $R_n = 0.5$ & $\Delta = 0.01$ m 100

5.1 Illustration of 3D QIM-based data hiding, here axis representation is in LiDAR frame 107

5.2 Block diagram of the proposed QIM-based framework 109

5.3 Attack models and tamper detection and localization results 114

5.4 Bounding box estimation of a ground truth label at different QIM-embedding step sizes 117

5.5 Bounding box distortion analysis for different bit-embedding schemes under Uniform additive noise attack 121

5.6 Bounding box distortion analysis for different bit-embedding schemes under Gaussian additive noise attack 122

5.7 Bit error rate of decoded code book for different step sizes and added uniform noise . . . 125

5.8 Bit error rate of decoded code book for different step sizes and added Gaussian noise . . . 126

5.9 Bounding box distortion in meters for different step sizes and added uniform noise . . . 128

5.10 Bounding box distortion in meters for different step sizes and added Gaussian noise . . . 129

5.11 Illustration of quantization noise . . . 131

5.12 Sequence diagram of proposed method . . . 133

5.13 Voxel centeroid movement due to dither modulation . . . 134

5.14 Message embedding example with three bits and L= 6 . . . 135

5.15 Bit Error Rate at $\Delta = 5cm, \sigma = 0.0$ . . . 138

5.16 Bit Error Rate at $\Delta = 5cm, \sigma = 0.0072$ . . . 138

5.17 Bit Error Rate at $\Delta = 5cm, \sigma = 0.0144$ . . . 139

5.18 Localization distortion at $\Delta = 5cm, \sigma = 0.0$ . . . 140

5.19 Localization distortion at $\Delta = 5cm, \sigma = 0.0072$ . . . 140

5.20 Localization distortion at $\Delta = 5cm, \sigma = 0.0144$ . . . 140

5.21 Detection False Negatives at $\Delta = 5cm, \sigma = 0.0$ . . . 141

5.22 Detection False Negatives at $\Delta = 5cm, \sigma = 0.0072$ . . . 141

5.23 Detection False Negatives at $\Delta = 5cm, \sigma = 0.0144$ . . . 142

6.1 Data-transactions in modern-vehicles and watermarking framework proposal . . . 147

6.2 OSI model of an Ethernet frame . . . 150

6.3 Block-diagram of the system model . . . 152

6.4 Spectrograms of sensor transmissions generated using 8 ms window size, 25% overlap, and Hanning weight window 154

6.5 Power spectrum of sensors under test at 25 cm distance measurement 156

6.6 Power spectrum of all sensors under test at different distances . . . 159

6.7 Data collection set-up for fingerprint extraction 160

6.8 Saturation of received signal - Spectrogram visual 161

LIST OF TABLES

TABLE

4.1 BER and False-Alarm rate at different noise levels 99

5.1 QIM-induced distortion at different step-sizes 116

5.2 VoxelNet: Car detection average precision scores 118

5.3 False alarm rates at different step-sizes for added noise 130

6.1 Accuracy: Gaussian NaiveBayes Method 161

6.2 Accuracy: Saturated Gaussian NaiveBayes Method 162

LIST OF APPENDICES

APPENDIX

A. Algorithms 166
1 Find Modified Indices 166
2 Find Deleted Indices 167
3 Find Added Indices 167
4 Watermark Sequence Generator 168

B. Source Code 169

LIST OF ABBREVIATIONS

ADAS Advanced Driver Assistance Systems

AD Automated Driving

AV Autonomous Vehicle

BER Bit Error Rate

CAN Control Area Network

CAN-FD Control Area Network - Flexible Datarate

CPS Cyber Physical System

DCT Discrete Cosine Transform

DFT Discrete Fourier Transform

DWT Discrete Wavelet Transform

ECU Electronic Control Unit

ECC Error Correction Code

EKF Extended Kalman Filter

GPS Global Positioning System

LiDAR Light Detection and Ranging

LSB Least Significant Bit

MAC Message Authentication Code

QIM Quantization Index Modulation

RADAR Radio Detection and Ranging

SecOC Secure Onboard Communication

SVD Singular Value Decomposition

V2X Vehicle to Anything Communication

WSN Wireless Sensor Networks

ABSTRACT

Sensors are used in multiple applications that touch our lives and have become an integral part of modern life. They are used in building intelligent control systems in various industries like healthcare, transportation, consumer electronics, military, etc. Many mission-critical applications require sensor data to be secure and authentic. Sensor data security can be achieved using traditional solutions like cryptography and digital signatures, but these techniques are computationally intensive and cannot be easily applied to resource constrained systems. Low complexity data hiding techniques, on the contrary, are easy to implement and do not need substantial processing power or memory. In this applied research, we use and configure the established low complexity data hiding techniques from the multimedia forensics domain. These techniques are used to secure the sensor data transmissions in resource constrained and real-time environments such as an autonomous vehicle. We identify the areas in an autonomous vehicle that require sensor data integrity and propose suitable watermarking techniques to verify the integrity of the data and evaluate the performance of the proposed method against different attack vectors. In our proposed method, sensor data is embedded with application specific metadata and this process introduces some distortion. We analyze this embedding induced distortion and its impact on the overall sensor data quality to conclude that watermarking techniques, when properly configured, can solve sensor data integrity verification problems in an autonomous vehicle.

CHAPTER I

Introduction

1.1 Introduction

In the modern world, sensors and sensor networks are used in many mission-critical applications. Different applications such as industrial control systems, healthcare, military, intelligent transportation, IoT (smart homes, smart infrastructure), etc. use sensors and rely on their data. Sensors measure the physical quantities from their environment and often convert them into measurable electric signals (Shin et al., 2016).

Industrial and infrastructure control systems rely on sensors and computer-based systems to monitor the physical processes. These systems, also known as process control systems, are used to connect the networked IT infrastructure to the physical world through sensors. These Supervisory Control and Data Acquisition (SCADA) systems or the Cyber-physical Systems (CPS) with the embedded sensor and actuator networks control several safety-critical applications. SCADA systems, in particular, perform vital functions in national critical infrastructures, such as electric power distribution, oil, and natural gas distribution, water and waste-water treatment, and transportation systems. These control applications can be considered safety-critical since their failure can cause harm to the physical system and even to the people who depend on those services (Cárdenas et al., 2011). The security of these safety-critical

systems depends on the integrity of the data sensed and transmitted by the sensors. These industrial control systems have to deal with many legacy components and interfaces where it becomes challenging to secure all the weak links. Many lightweight cryptographic mechanisms like IEEE P1711 standard to secure legacy serial links were developed to ensure data integrity and confidentiality in these systems, yet, studies show that to secure the critical control systems properly, the underlying technology must satisfy some minimum performance requirements to allow the implementation of well-tested security mechanisms and standards. In the absence of such infrastructure, alternate technologies like watermarking- based sensor security methods need to be researched. With the recent advances in IoT, wireless sensor networks, and their penetration into industrial applications, the problem to check for sensor data integrity becomes more crucial and challenging.

In the recent episode of the theft of RQ-170 Sentinel UAV (unmanned aerial vehicle), one of the popular theories on the technology behind this controversial theft emphasizes the importance of checking for sensor data integrity. As per this theory, the UAV was spoofed to land into an enemy airfield through a GPS spoof attack, where-in a GPS satellite signal is overlaid by a spoofed GPS signal from a local transmitter, which lead to the errors in position estimation of the UAV (Hartmann and Steup, 2013).

Body sensor networks (BSN) is a recent advancement in healthcare management that relies on the power of the internet of things (IoT) to bring the patients closer to physicians. BSN lets physicians collect data round the clock from the patient using low-power and lightweight wireless sensors that monitor the patient and his environment. These networks are widely used in senior citizen healthcare, where many sensors wearable, implanted, and environment are used to enable aged people to enjoy new medical healthcare services ubiquitously. The data collected over these networks is sensitive both from the privacy and integrity perspectives. Since this data

collection does not take place in a controlled environment, the process needs strict security mechanisms to prevent any malicious attacks in the form of data tampering. Data integrity is one of the key elements along with the privacy, authentication, and anonymity of BSNs. Data integrity can be defined as protecting data from external modifications. In the case of a BSN or any sensor network, an adversary can always alter the data by adding some fragments or by manipulating the data within a packet. In the case of life-critical applications, the lack of a mechanism to detect this data manipulation could become dangerous (Gope and Hwang, 2016).

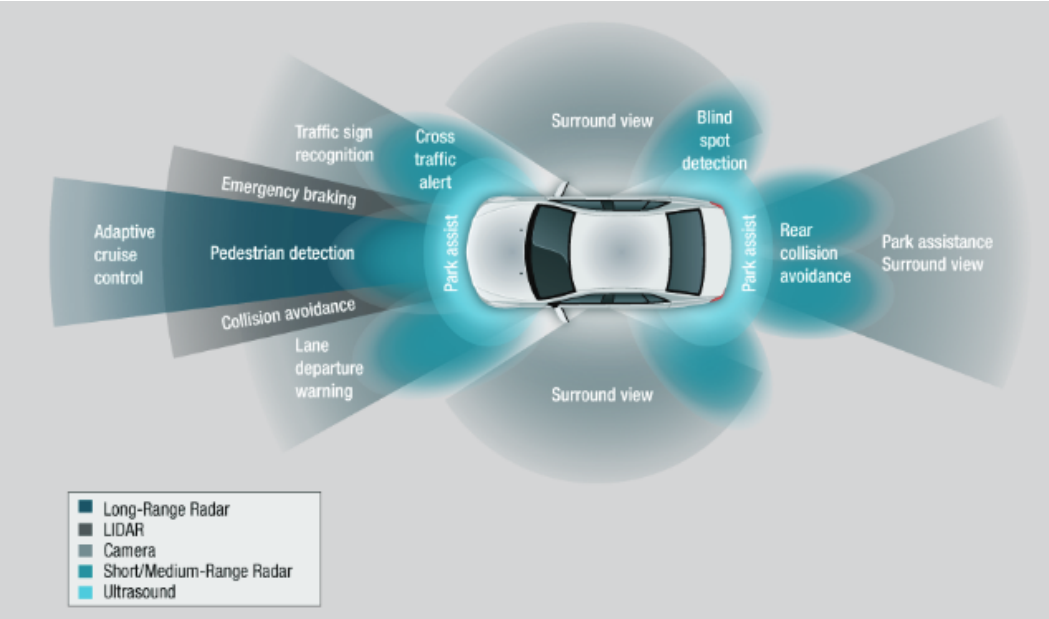


Figure 1.1: Autonomous vehicle sensor suite (Snehaprabha and Ram, 2019)

An autonomous vehicle or a self-driving car is a disruptive technology altering the future of ground transportation. As per the European Commission report in 2017, the revenues from autonomous driving are expected to increase from 7.6 bn in 2015 to 50 bn EUR in 2022 (European Commission, 2017). Modern car as a cyber-physical system heavily relies on sensors. Starting from the biometric sensors like fingerprints scanners, voice recognition sensors (microphones) that give the user access to a vehicle to the sensors which help autonomous vehicles drive by themselves like

LiDARS, RADARs, cameras, and ultrasonic sensors, etc. a modern car relies heavily on the sensor data. There are five essential functions that an autonomous car needs: perception, localization, planning, control, and system management. Perception is the process of sensing the surrounding environment of a vehicle(Jo et al., 2014). The typical sensor suite used to build the perception layer in an autonomous vehicle is depicted in Figure 1.1. The most commonly used sensors are cameras, RADAR, ultrasonic sensors, and LiDAR. The main challenge with the current sensor suite of an autonomous car is that every sensor has its own deficiency. Some work better in different weather conditions like rain and snow, and others work better in different lighting conditions. If each sensor is considered individually, it cannot cover the wide spectrum of vehicle operating conditions and environments. For example, cameras can be spoofed by glare, RADARs have a very narrow field of view, ultrasonic sensors can be only used for near field detections, and LiDARs cannot work well in rainy conditions. Hence, the concept of sensor fusion is developed where the perception layer is generated after combining multiple sensor outputs. As the concept of autonomous vehicles is getting close to reality, engineers are in a quest to make their systems more safe and secure and in search of new sensors like infrared cameras, ground-penetrating RADARs (Quain, 2019). This dependency of an autonomous vehicle on sensors and their data puts forward the challenge of data integrity and brings us to the question of whether this sensor data is intact while passing through a vehicle network. Given any application domain discussed above, to ensure the correct operation of the actuation and control systems that depend on the sensors their data must be authentic and robust to spoofing.

1.2 Attacks on Sensors

Sensor data spoofing can be broadly classified into three categories (Shin et al., 2016)

1. **Regular channel**: Direct attacks on the physical sensing structure, such as spoofing a microphone using a sound wave (Shin et al., 2016). Jamming attack on an ultrasonic sensor using an ultrasound generator operating in the same frequency as the sensor (Chen et al., 2016).

2. **Transmission channel**: Attacks on the sensor output transmission channels (wired/wireless), such as frame tampering of LiDAR sensor data (Changalvala and Malik, 2019b).

3. **Side channel**: Attacks on the sensing structure by using a different physical system, such as using light to inject malicious attacks on the voice-controlled devices (Sugawara et al., 2020).

In many modern and legacy systems, having end-to-end security of sensor data from these three kinds of attacks is very crucial. In an era where smart devices powered by sensor data are driving our cars, influencing our purchase decisions, automating our homes, and providing us with continuous connectivity, the need to secure the sensor data takes higher priority. This need of the day has been a major motivation factor in our research. In this research, we focus on securing sensor data from the transmission and regular channel attacks in resource-constrained systems such as an autonomous vehicle. In the future, it can be extended to a broader scope of end-to-end data transaction security in autonomous vehicles and other domains.

1.2.1 Automotive Sensors Vulnerability

In the last 20 years, the automotive industry has seen rapid growth in the usage of electronic control units (ECUs) to implement various technology features such as dynamic vehicle controls, infotainment, and ADAS. These technologically enhanced user experience and safety features are heavily dependent on the sensors, and advanced communication networks integrated into the vehicles. An autonomous vehicle (AV),

in particular, depends heavily on sensors and data transmission networks to implement the highly automated driving functions (Sarmento et al., 2017). Autonomous vehicles rely entirely on sensors to estimate their surroundings, to detect and react to obstacles. To achieve a sustainable Society of Automotive Engineers (SAE) level 2 automation and above (SAE Ground Vehicle Standard), a typical vehicle is equipped with multiple sets of sensors, like cameras, LiDARs, and RADARs, etc. To get a perspective, the GM Cruise AV is equipped with 5 LiDARs, 16 cameras, and 21 RADAR sensors (Baxter et al., 2018). An autonomous vehicle internal communication network is widespread and relies on multiple physical interfaces such as Controller Area Network (CAN), CAN-Flexible Data rate (CAN-FD), Ethernet, Local Interconnect Network (LIN), FlexRay, etc. These in-vehicle networks are interconnected over gateway modules transmitting data and control commands. The sensor data flows through the vehicle network to the centralized data processing unit called the Advanced Driver Assistance System (ADAS) module or the vehicle's brain from the sensors mounted on the vehicle.

The vehicle internal network topology, which consists of ECUs, gateways that forward data from one interface to other, and the connectivity to the external world over cellular and other wireless interfaces make a modern vehicle a cyber-physical system vulnerable to cyber attacks (Cui et al., 2018). In this distributed architecture, it is possible to inject code through available attack surfaces like the onboard diagnostics port (OBD-II) and CAN bus into the core ECU and bridge across multiple networks, thus exposing attack surfaces on different networks.

Numerous attack vectors have been proposed in detail for the CAN and the OBD-II port over the past decade. It was demonstrated that attackers can infiltrate any ECU and circumvent safety measures to modify the outcome of safety-critical systems, disable brakes, perform steering control, or cause faulty cluster displays and even a complete engine shutdown (Checkoway et al., 2011). The shortcomings in the CAN

protocol such as the broadcast message format, clear data transfers, and lack of mechanisms to establish data authentication and confidentiality, expose the CAN network and the nodes connected, such as sensors to masquerade attacks, replay attacks, and other exploits (Lin and Sangiovanni-Vincentelli, 2012).

In a connected vehicle, attacks could be launched over wireless channels without physical access to the vehicle (Wyglinski et al., 2013). In-vehicle sensor data communication is unencrypted, and therefore an attacker just needs access to the in-vehicle network for sensor data manipulation, which can be realized through available attack surfaces. Hackers can exploit these attack surfaces for sensor data manipulation. In a given vehicle with autonomy levels higher than SAE level 2 (semi-autonomous mode), a cyber-attack on the sensor data could lead to life-threatening accidents as the driver would be disengaged partially or completely trusting the system (Liu et al., 2017). Consider the example of an Ultrasonic sensor. These sensors are most commonly used in modern vehicles for close-range object detections and park assist features. Tesla Model S relies on ultrasonic sensors to achieve its "Smart Summon" feature. When an ultrasonic sensor is attacked employing jamming and spoofing, it can perceive an object that is not truly there (false positive), or cause the sensor not to perceive an object that truly is there (false negative) as shown in (Xu et al., 2018). These incorrect readings can cause damage to a vehicle, building, or even human life. It has been successfully demonstrated in (Chen et al., 2016) that these sensors are vulnerable to attack and can have fatal outcomes. Hence, securing sensor data transmissions becomes a very critical component for the safe functioning of an autonomous vehicle (Longxiang et al., 2017).

1.2.2 Automotive Sensor Categories

Autonomous vehicle sensors are broadly divided into two categories as smart and raw sensors based on their data processing capacity. Raw sensors are the ones that

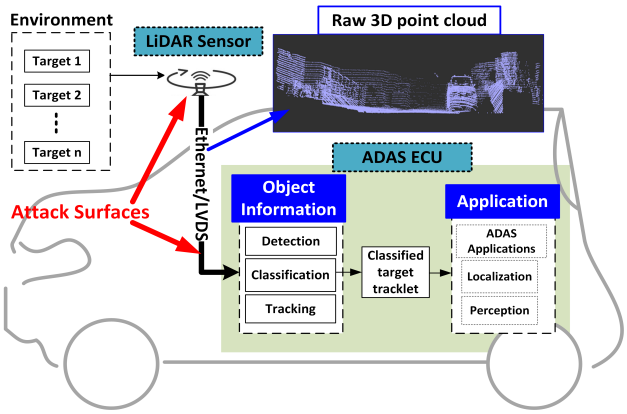


Figure 1.2: Autonomous vehicle sensor suite - Raw sensor

transmit the raw or unprocessed data to the processing unit inside the vehicle. Shown in Figure 1.2 is a high-level illustration of (a) data flow from various raw sensors to the sensor fusion core residing in a vehicle, also known as ADAS (advanced driver assistance system), and (b) available attack surfaces. For instance, a LiDAR sensor mounted on the vehicle sends a raw point cloud of the tracked environment over an Ethernet/LVDS link to the sensor fusion ADAS core ECU placed in the vehicle. Inside the sensor fusion core, the object information from the raw sensor data is extracted and a list of tracked-objects called tracklets is computed and provided as an input to the perception estimator and other applications like the vehicle localizer. The raw sensors, specifically the ones that have high data rates such as the camera and LiDAR use the Ethernet interface for their data transmissions to the ADAS unit. Smart sensors are the ones that can detect and track objects internally, send the tracked object list to the vehicle over limited bandwidth interfaces such as CAN/CAN-FD. Figure 1.3 shows data flow from smart sensors to the sensor fusion core residing in a vehicle and the available attack surfaces. For instance, a RADAR smart sensor mounted on the vehicle tracks the targets and extracts the object information from the raw sensor data to build a list of tracklets. This information is provided over the low bandwidth interfaces like CAN/ CAN-FD to a perception estimator and other applications running inside the ADAS unit. The majority of RADAR and Camera

sensor sets used in autonomous vehicles are smart sensors. The data from these smart sensors is fused inside the vehicle ADAS unit to determine the final list of surrounding objects (Jo et al., 2015). The decision to select a smart sensor over a raw sensor is driven by the autonomous vehicle architecture, functional safety and redundancy requirements. In most architectures, a combination of both types of sensors is considered.

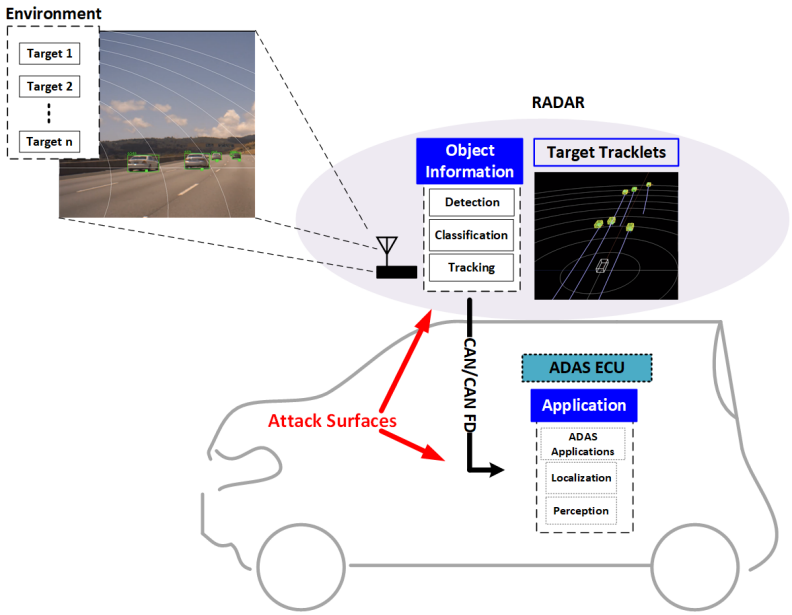


Figure 1.3: Autonomous vehicle sensor suite - Smart sensor

1.3 Problem Statement

Like any other cyber-physical system, an autonomous vehicle is vulnerable to internal attacks on the sensor networks. The 2015 Jeep hack by Miller and Valasek where they could bring the vehicle to ditch through a remote connection highlights the risk of cyber attacks on modern connected vehicles. The impact of such attacks is more in vehicles supporting driver assistance and automated driving features. Hackers can take advantage of the feature implementation flaws, code bugs, etc. to launch remote or insider attacks. Imagine a typical ride-share application use case of an autonomous

car, hackers have a direct access to the vehicle during the ride without any human supervision, giving them ample time to meddle with the system and sensors. It is a need of the hour to double-check the integrity of the sensor data in autonomous vehicles before processing and acting on it. But, before searching for options to protect the sensor data, let us first take a look at some of the constraints of the automotive sensor networks. This will help us to come up with practically implementable solutions. Sensor networks in autonomous vehicles have two constraints.

1. **Limited bandwidth**: The communication interface from the sensor to the data processing unit is bandwidth limited in automotive applications. Most sensors use a traditional CAN interface with an 8-byte payload, which restricts the usage of traditional cryptographic methods to secure the sensor data (Zou et al., 2017). An enhanced version of CAN called CANFD is introduced to increase bandwidth and payload to up to 64 bytes. CANFD allows AUTOSAR secure onboard communication protocol (SecOC) implementation on the network. Apart from issues like key management, time synchronization, the SecOC requires the transmission of a message authentication code (MAC), which can take up to 8 bytes of the payload space. Given a scenario where multiple sensors are connected to the same network and increasing demand on the sensors to publish more data to build a high-resolution perception, bandwidth can become a constraint even in high throughput interfaces like CANFD.

2. **Real-time data inference**: Autonomous vehicle applications often require the sensor data to be processed in real-time. This constraint makes it difficult to use traditional data security methods based on cryptography as they require an additional step of decryption before data becomes useful.

With these two constraints in mind, to design a framework that can work better under these conditions, we researched the state-of-the-art in sensor data integrity verification

methods. In our study, we identified that wireless sensor networks (WSN) share similar constraints as autonomous vehicle sensor networks preventing them from using any traditional cryptography methods and significant research was done on securing the data transfers in WSN using watermarking methods. Hence, in our literature review, we focused on understanding the state-of-the-art watermarking techniques implemented in wireless sensor networks.

WSNs are extensively used in many fields like environmental monitoring, military surveillance, traffic monitoring, patient monitoring, etc. In WSNs, the underlying assumption is that the sensor nodes collect information and send it to a central node for processing over a wireless interface. WSNs have some typical characteristics when compared to other traditional networks. They do not care about the origin of the data. Their primary focus is to collect perception data and to transmit it to the destination. The sensor leaf nodes are typically low capacity lightweight processing units, and the networks reconfigure dynamically based on the sensing environmental conditions. This limitation creates a challenge to design a security mechanism that is both secure and energy-efficient. The dynamic nature of these networks makes them susceptible to adversary attacks such as data tampering, forgery, selective forwarding, replay, and transfer delay attacks (Zhang et al., 2017). The biggest problem these networks face is how to secure the data without increasing the burden on the leaf nodes as they are usually resource-constrained. Traditional data integrity authentication methods such as cryptography and message authentication code (MAC) cannot be applied to these networks. Encryption algorithms require additional key storage space and computation power which in-turn increase the energy consumption and storage requirements of the sensor leaf node. Also, though the encrypted data is safe, the data can only be used after decryption, and the decrypted data could again become a target of attackers. To solve these issues with traditional methods, several watermarking techniques were researched for WSN applications. Using watermarking

for sensor integrity checks started in 2003 with the Feng et al. (Feng and Potkonjak, 2003) proposal of embedding cryptographically encoded signatures into the data payload. From the literature review, it is evident that the watermarking techniques were applied to wireless sensor networks with applications in IoT and healthcare, etc. To the best of our knowledge, watermark generation based on the sensor data characteristics and applying it to check the data integrity in resource-constrained and real-time sensor networks of an autonomous vehicle is the first of its kind effort. This is the primary focus area of our research. We want to identify and analyze applications where the data hiding techniques could be of help in the autonomous vehicle domain.

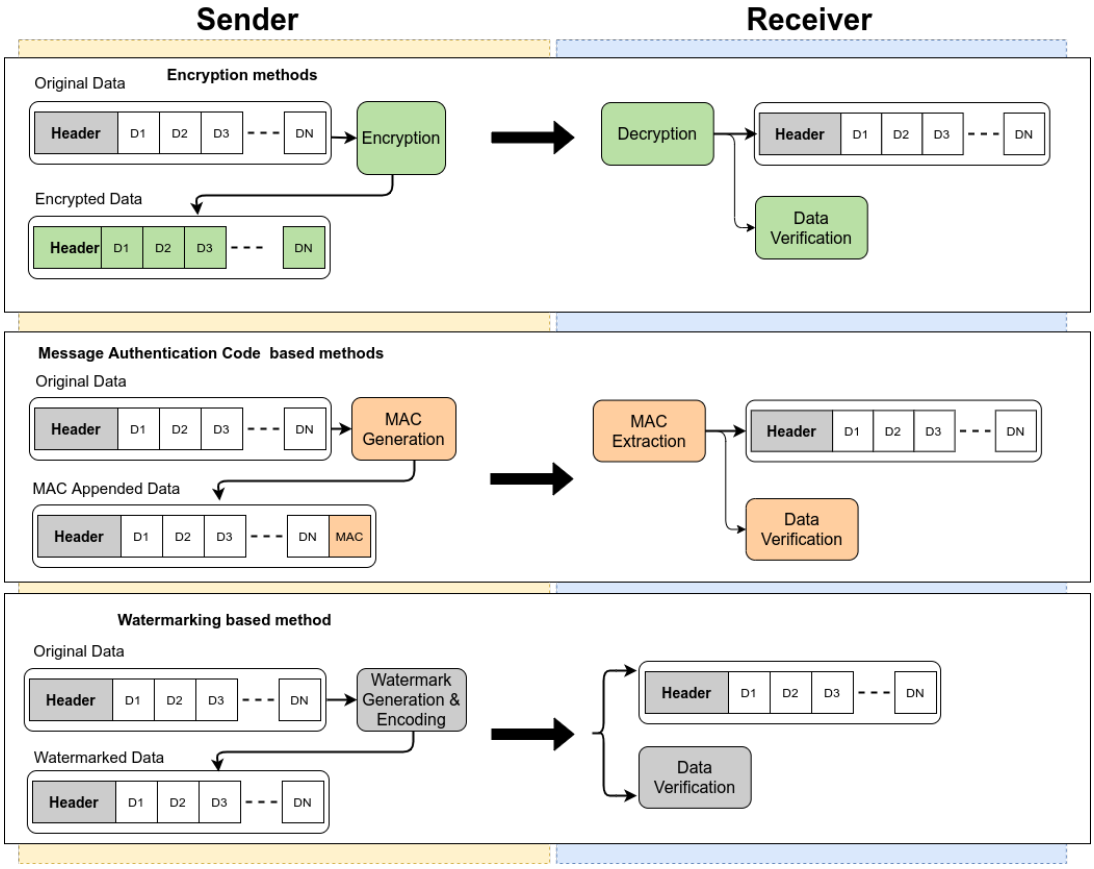


Figure 1.4: Comparison of different methods to achieve sensor data integrity

1.4 Watermarking Advantages

In this digital age, with the growing trend of multimedia information exchange, the need to ensure data confidentiality and integrity is increasing. Data transfers over different networks are vulnerable to various attacks such as privacy infringement, content stealing, and tampering. The concept of network information security deals with securing the data transmissions, storage, and processing from data leakage, theft, tampering, and deletion. To ensure the network information security, traditionally cryptography based methods were used. Cryptography ensures communication privacy, data confidentiality, and authentication by encrypting the data using different key sharing mechanisms. Though cryptography can be used to solve information security issue, there are some shortcomings to that approach. Encryption can draw the attacker's attention towards sensitive information and motivate them to crack it. Once the attackers break the encryption, they have complete access to the data. Even if the attacker fails to crack the encryption, he can slightly modify the data to make the entire transaction invalid. Data-hiding-based watermarking techniques started gaining attention over the past two decades particularly in the areas that require the prevention of unauthorized access to confidential information. These methods do not reveal their existence in the data, thus providing more protection than cryptography in some applications. Data hiding methods also differ from traditional cryptography in their purpose. The main objective of data hiding is not to restrict normal access to the data but to ensure that the embedded secret information is not violated or discovered (Lu and Guo, 2017). This embedded message can be used to track the data forgery through different data transactions and also to localize it. The cryptography methods are usually complex and computation resource-hungry, hence they cannot be applied every-where. In the edge computing devices such as an autonomous vehicle, where the computation resources are a big constrain, cryptography methods cannot be applied. There are also other significant drawbacks of using cryptography

methods in edge systems, like key management burden, network bandwidth issues, and export restrictions on the products using specific cryptography methods. These complexities increase both the production and maintenance costs of the features and products. In an autonomous vehicle, consider a scenario where a couple of satellite RADARs are connected over a CAN network to a decision-making unit that does the sensor data fusion to implement an ADAS feature. To implement sensor data integrity verification using traditional cryptography techniques requires modifying each smart RADAR sensor to include a hardware trust anchor to store the keys and accelerators to verify the signatures. This increases the RADAR product cost and any MAC-based security mechanism will also increase the payload overhead, therefore requiring a different interface other than CAN that supports higher bandwidth, which again is not feasible in legacy systems.

In comparison to conventional cryptography, watermarking methods are computationally less complex and less resource hungry. Data modified by the watermarking algorithms can be directly used by the end application without having to modify or to clean it before usage. This comes as an advantage for applications that rely on real-time data processing. In Figure 1.4, different methods that can be used to protect and verify the data integrity in an AV are compared. In a sender-receiver scenario, to protect the data integrity, one can use either encryption, MAC embedding, or watermarking. In both encryption and MAC-based methods on the receiver end, applications cannot use data until it is decrypted or the MAC is removed. This additional computational step can become a bottleneck in the applications that require real-time data processing (Changalvala and Malik, 2019b). In such applications, a cryptographic encryption step would make the data useless until the data gets decrypted, adding to the processing delays.

In MAC-based methods, the data payload increases dramatically based on the interface bandwidth. In most of the legacy systems discussed in section 1.1, the

bandwidth of the interface between the sensor system and the data processing unit is a severe bottleneck. This is one of the reasons why the automotive industry was not able to implement any cryptography over the controller area network (CAN) interface (Zou et al., 2017). For CAN interface in particular, this method is not recommended as it could double bandwidth requirements (Woo et al., 2016), (Zou et al., 2017). Data hiding based watermarking works directly on the host data by perturbing the data by a negligible amount, thus eliminating the need to increase the data payload and network bandwidth. As shown in Figure 1.4, all the three methods have a data integrity verification step in common that can be separated as an independent process and run in parallel to the algorithms that process the data. Again, in the case of watermarking, this data verification step does not require much computation resources, unlike cryptography.

Watermarking also provides much-needed data traceability. The security offered by watermarking does not stop at the application level. Watermarked data provides security beyond the autonomous driving application since the data cannot be stripped of any additional payload. The embedded watermark stays with the data until the information is used by the consumer application and beyond. Applications acting as a pass-through to the data such as an on-board data recorder as-well-as secure logging mechanisms that push the sensor data to the cloud for analytics can benefit from the watermarking of the data, this concept is discussed further in chapter 6. When data gets exchanged from one entity to another and security keys are shared, end to end encryption does not help in identifying the leakage points whereas watermarking can. These advantages make data-hiding based watermarking techniques a better choice over cryptography in many edge computing applications such as sensor data integrity verification in autonomous vehicles. One aspect to consider while using the data hiding or watermarking techniques is the embedding induced distortion. These techniques cannot be blindly applied to applications where the accuracy of the data

is critical over integrity. The embedding included distortion and its impact on the application need to be studied and watermarking intensity need to be adjusted to fit a specific application.

1.5 Thesis Outline

In this research, we propose a framework to implement data integrity verification in autonomous vehicle networks using watermarking techniques. Our research aims to

1. **Identify the applications that are vulnerable to cyber attacks due to lack of sensor data integrity checks**: Analyze the security risks in autonomous vehicle applications in particular. Here, two types of applications are targeted

    - Resource constrained applications: Bandwidth and power-constrained
      Autonomous vehicle use case: Securing RADAR tracklet data transmitted over CAN network (Nabati and Qi, 2019), (Changalvala et al., 2020),
    - Time constrainted applications: Real-time data processing
      Autonomous vehicle use case: LiDAR sensor raw data object detection and recognition (Changalvala and Malik, 2019b), (Changalvala and Malik, 2019a)

2. **Propose a watermarking or data hiding based solution**: Identify data hiding based watermarking solution to the problem at hand. Suggest low complexity watermarking based methods to secure sensor data, independent of the transmission interface.

3. **Test the solution with real-world data**: Our approach is to test the pro-

posed framework on publicly available industry-standard datasets. In the case of self-driving car applications, datasets such as the KITTI vision benchmark suite, Mercedes Benz autonomous driving utility are used to evaluate the performance of the proposed method. The evaluation criteria includes computational complexity, robustness, accuracy in detecting the tamper and embedding induced distortion estimation, etc. The distortion estimation is done on the vehicle dynamics models and deep learning algorithms used extensively in developing autonomous driving algorithms.

1.5.1 Document Road Map

- Chapter 2 "Data Security Using Digital Watermarking". In this chapter, we provide a brief overview of watermarking techniques starting with their origin and history. Discuss various applications of watermarking. Give a perspective of watermark security aspects like securing the watermark vs securing the data and outline different attacks on watermarks. Provide key differences between watermarking and cryptography. Discuss various design requirements of watermarking techniques and their mathematical representations. We conclude the chapter by discussing the state-of-the-art watermarking techniques and provide some examples of the techniques used in this research.

- Chapter 3 "Watermarking & Data Models". In this chapter, we define models that represent sensor data and demonstrate how the sensor data can be viewed as $n-$ dimensional vectors in R^n. Explain the watermarking concepts using geometrical and communication model analogies. Transition into scalar watermarking techniques like the Quantization Index Modulation (QIM) watermarking method and explain the need for high dimensional lattice QIM codes.

- Chapter 4 "RADAR Data Integrity Verification-2D QIM". In this chapter, we

discuss the implementation of a smart sensor data integrity verification method. We propose a 2D QIM method to embed the watermark into the position information of a RADAR sensor. We propose algorithms to detect and localize the attacks on the integrity of the RADAR sensor data. The embedding-induced distortion and its effects on sensor fusion algorithms are tested using an Extended Kalman Filter-based motion model and the results are discussed.

- Chapter 5 "LiDAR Data Integrity Verification-3D QIM". In this chapter, we discuss the details of securing the raw sensor data using a LiDAR sensor. We propose a 3D QIM technique to secure the raw point cloud obtained from the LiDAR sensor and analyze the effects of embedding induced distortion on the deep learning models performing object detection on the LiDAR data. We also discuss the need to extend the plain QIM to a more random spread spectrum dither QIM method and identify the optimal configuration values for this method.

- Chapter 6 "Future Work & Conclusion ". In this chapter, we propose future directions to the research. We discuss how sensor fingerprints are used to build countermeasures to regular channel attacks. We propose ideas to develop an universal data transaction traceability and integrity verification mechanism for various data transactions in autonomous vehicles.

CHAPTER II

Data Security Using Digital Watermarking

2.1 Watermarking Background

The reference to the use of watermarks dates back to 480 B.C in Greek mythology. In a popular tale of Herodotus, Histiaeus uses an ingenious watermark technique of engraving a tattoo of the secret message on a slave's shaved head and conceals it by regrowing the hair. The slave reveals the secret message to Miletus after shaving the head again. This explains the core of the watermarking where the secret message can be communicated in plain sight. Along with the history of humankind, many techniques of watermarking were portrayed especially for wartime communications. The commercial applications of watermarking such as copyright protection can be seen in 17th and 18th-century logarithmic tables. In these tables, errors were introduced into the least significant bits to protect the intellectual property (Moulin and Koetter, 2005). In the modern era, with the advent of the internet and communication networks, many applications started falling prey to data leaks and copyright infringements thus increasing the need to develop techniques to counter such attacks. Digital communications are increasingly getting susceptible to adversary attacks with the development in computer technology and internet connectivity and so is the need to ensure data confidentiality and integrity. People can now use the internet to exchange data making it easy to both infringe the digital rights and also launch malicious at-

tacks on confidential data. With the pervasive usage of the internet came the concept of network information security. It deals with securing the data transmissions, storage, and processing from data leakage, theft, tampering, and deleting (Lu and Guo, 2017). The increase in digital content in the form of digitized books, music, and videos brings forward the challenge of copyright protection where the content producer rights need to be protected as this digitized content is transmitted over the network and shared by end-users. This openness in resource sharing needs efficient methods to detect and prevent the tampering, plagiarism, and embezzlement of the data. Several data protection mechanisms were developed over the last few decades to solve this problem. These methods can be used independently or in combination based on the problem they are solving. Encryption methods that are based on the symmetric, asymmetric keys and hash functions are traditionally used to solve data integrity issues. Message authentication can be performed using a digital signature encryption scheme. These cryptography methods have some disadvantages such as indicating attackers that the encrypted message is important, loss of security once the attacker figures out the encryption method, giving the attacker a chance to make the information fail authentication and make it useless with a slight change in the data (Lu and Guo, 2017). In the late 90s, researchers started investigating alternate methods to address some of the shortcomings of the cryptography methods and there has been a revived interest in watermarking and data hiding-based methods. In the following sections, we try to cover multiple concepts of watermarking starting with the most common applications.

2.2 Applications of Watermarking

Digital watermarking is the art of embedding digital data into various forms of digital media such as images, audio, video, and other digital object data. Traditionally digital watermarking applications have been in the digital media domain but

lately, these techniques are extended to other fields such as medicine and sensor networks. Based on the properties of the watermark various applications can make use of them. Discussing the watermarking properties further would help us understand the requirements of the watermarking better. Watermarking the digital data or object data, in general, provides a way to measure the data

1. Authentication
2. Confidentiality
3. Integrity
4. Access control

These properties make watermarking techniques apt for many applications such as

1. Content Identification & Protection
2. Digital forensics
3. Ownership and copyright protection
4. Content security and authentication
5. Broadcast monitoring
6. Online content location
7. Improved auditing
8. Access control
9. Clandestine communications

2.2.1 Content Identification and Protection

Content identification helps the digital media copyright owners, brands, and distributors to understand when and where their digital content is being consumed. It can deter unauthorized use and help with an audit by reporting the distribution paths of the content with accuracy. At the same time for the consumers, it gives an ability to seamlessly stream content across multiple devices, makes the content easy to search, and enables parental controls. Digital watermarking provides methods to implement content identification management by providing a unique digital identity to the media content that can persist even when the content gets transformed. Digital watermarks can be easily embedded into the content without interfering with consumer enjoyment. The embedded watermark is imperceptible by humans but can be easily identified by digital devices (Digital Watermarking Alliance, 2020).

2.2.2 Digital Forensics and Piracy Detection

In this application the watermarking gives an ability to the content owner to detect and respond to the misuse of the assets, to gather evidence in criminal proceedings, and also to enforce the contractual user agreements. Watermarking will supplement digital rights management by combining the content owner copyrights with consumer fair allowances. The way it works is, a forensic application embeds the identity of a recipient into an asset copy at the time of production or transmission. The identity could be situational metadata such as time, receiving format, and receiver details such as an IP address. The forensic watermark retrieved from the leaked copy helps identify the intended recipient.

2.2.3 Ownership and Copyright Protection

Copyright protection deals with proving the origin of the digital content. This is the most common application of watermarking. The basic idea is of being able to

identify the owner of the product by embedding some information. If the product's origin generates a controversy the embedded watermark can be extracted to determine the owner if it.

2.2.4 Content Security and Authentication

The impact of counterfeiting digital content is widespread in terms of lost revenues to the producer and fraud to the consumer. With digital watermarking, the content can be authenticated through imperceptible watermarks embedded into the digital content. The watermark can be encrypted and secured to further fortify the security and only let the authorized devices to detect and access the data.

2.2.5 Location of Content Online

For the content producers, the need to protect their digital assets arises in today's growing content sharing platforms. In the corporate world as well the documents and videos are transmitted through emails and across the internet. In these scenarios the content producers should be aware of how the consumer is making use of their content and if it is up to date. Digital watermarking can help through imperceptible E- digital IDs. These IDs which are identified through unique search engines can be used to generate reports and notify the owner of the location of the content.

2.2.6 Broadcast Monitoring

This application of watermarking, checks if a particular content is airing or not. With the increase in the number of broadcast stations this ability to monitor content broadcasting becomes important. Watermarking the content at the time of content production or broadcast allows the broadcasters with great precision when and where the content has been broadcasted and for how long. Digital watermarking works by making subtle modifications to the original data and by adding some bits of data

disseminated through the content.

2.2.7 Auditing

In a simple application of watermark usage in auditing, content owners and distributors may embed unique digital watermarks which serve as an identifier for each licensed asset in their content. The identifier remains embedded in the asset and is transmitted to every composition of the licensee that comprises all or part of the original asset. This can enable an auditing application to evaluate the use of the owner's assets from every composition easily and automatically.

2.2.8 Access Control

Access control can be ensured by introducing client-side software that blocks relevant media content based on the presence or non-presence of a watermark. TV broadcasting, access to medical records, and network architecture, etc. are some applications of access control.

2.2.9 Medical Applications

Medical records contain sensitive user information. Privacy, and integrity of which are of utmost importance. Watermarking the medical records helps in maintaining the access control, secured transmission, and prevents mismatch of the records.

2.2.10 Clandestine Communication or Steganography

Steganography is a form of conducting covert or stealth communications using watermarks. Steganography can be considered as a subgroup of watermarking since its goal is to embed a secret message into a medium. Steganography differs from traditional watermarking in the fact that it does not care about the medium of data exchange as long as the secrecy of the embedded message is maintained. In

watermarking the signal that gets watermarked is referred to as original work or a host signal and in Steganography, it is called cover work, these three notations are used interchangeably in this document. The three most desired characteristics of a steganographic system are undetectability, low watermark embedded distortion, and high capacity. Steganography finds many applications such as wartime communications, clandestine communications by journalists or whistleblowers to avoid censorship control, etc.

Data hiding techniques form a subset of watermarking methods that are extensively used in steganography. In data hiding, a message is embedded in digital media that can be retrieved later for establishing copyright or identification. Making the data hiding technique robust to the compression algorithms or other distortions is a prime priority in these applications. The embedded data need to be recovered and reconstructed if needed by the application even when the host signal gets distorted. The other main constraint of the data hiding technique is that the embedded data need to be directly encoded into the media than into any header or wrapper, and this embedding should not introduce a perceivable distortion to the host signal.

2.3 Watermark Security

Security requirements for the watermark can be viewed from two different perspectives. One accounts for the watermark itself when the attacker tampers the watermark and tries to defeat its purpose. The other accounts for the integrity verification of the original data into which the watermark is embedded. These security requirements of the watermark and the nature of the security are application dependent. In the applications that do not require securing the watermark the watermark is embedded to add value to the content and customer convenience rather than to restrict the content usage. In these cases, watermark security is insignificant since there is no motivation to tamper with the watermark (Cox et al., 2008). On the other hand, applications like

digital rights management might require the prevention of unauthorized embedding, detection, and removal of watermark along with the protection from various system attacks. These attacks that defeat the purpose of watermark embedding can be dealt with by combining cryptography with the watermarking schemes. For example, in the case of unauthorized detection, watermarking needs a large keyspace to secure against this attack where-as by adding straightforward cryptography to the solution this issue can be solved. The watermark message can be encrypted before embedding and can be decrypted after the detection. Also, to prevent unauthorized embedding, asymmetric key cryptography can be used to verify the source of the watermark embedding. Cryptography can also be used to verify if the watermark recovered belongs to the original work.

2.3.1 Threat Models

The watermarked signal can undergo several degradations in its lifecycle. Some of these degradations can be natural such as the additive white Gaussian noise added to the signal when it gets transmitted over a noisy channel or noise due to interference of signal attenuation specifically in wireless transmissions. In the case of transmissions over wired or wireless networks as the data passes through multiples nodes and gateways there are chances of network-induced data attenuation like packet loss. In some applications, the host signal has to go through certain required or legitimate manipulations like data compression, transcoding, and signal processing, or any other geometric manipulations. These changes to the data signal elements also affect the watermark embedded into the data. The other type of degradations that could affect the watermark are malicious attacks launched on the data either to compromise the data integrity or to erase the watermark. When it comes to watermark security and the host data security here are some of the common attack vectors that a typical watermarking model has to deal with (Artru et al., 2019). Also, even adversaries can

be active or passive. A passive adversary passively monitors the transmission channel and illicitly reads the messages whereas the active adversary tries to either disrupt the communication or modifies the data to transmit unauthorized data.

2.3.1.1 Removal Attack

In this attack the unauthorized user tries to remove the watermark or certain targeted content from the host data signal (Begum and Uddin, 2020).

2.3.1.2 Interference Attack

In these types of attacks, the watermarked signal is embedded with noise through signal processing functions like averaging, quantization, compression, etc. Here noise can also be considered as an unwanted modification of data.

2.3.1.3 Geometric Attack

In these attacks, the host signal is subjected to geometrical transformations such as translation, rotation of data points. In the case of images, this can also include operations like cropping, image rotations, etc.

2.3.1.4 Filtering Attack

In these types of attacks, the signal data is passed through filters to remove low or high-frequency components.

2.3.1.5 Active Attack

In the case of active attacks, the attacker tries to manipulate the watermark to make it undetectable or completely removes it by directly manipulating the host signal carrying the watermark at certain locations.

2.3.1.6 Passive Attacks

In these attacks, the attacker stays dormant sensing and observing the data for the presence of any watermarks

2.3.1.7 Data Degradation

In these types of attacks, parts of the host signal are removed that could not only affect the integrity of the signal itself but also the embedded watermarks. When it comes to protecting the original work, we need to answer several questions about its authenticity such as, if it has been altered mildly or significantly, what parts of the work have been altered, and can the work be restored. Although we might use cryptography-based approaches to get answers to these questions about the tampering, watermarking methods are still useful as they do not require auxiliary data and they undergo the same transformations as the host signal. To verify if the original work has been changed at all or slightly modified, the exact authentication method is used. This can be achieved by using fragile watermarking methods that are designed to become undetectable with the slightest change in the original work. Another approach could be embedding the cryptographic signatures as the watermarks. To verify if the original work is modified by a limited set of illegitimate distortions while allowing a predefined set of legitimate distortions is called selective authentication.

This can be achieved by using semi-fragile watermarking schemes. These methods are built to survive a predefined set of distortions but they get destroyed by any distortions beyond that set and hence deemed illegitimate. Tamper localization or the ability to identify the time or the area where the original work got tampered with is useful to verify the motive, source, legitimacy of tampering as-well-as to retrieve the unaltered data for the application to use. Many techniques are available to localize tampering, one of the main methods is the block-wise content authentication where the cover work is divided into disjoint temporal or spatial regions, and each of which

is authenticated separately. The accuracy of block-based localization depends on the size of the block and the smaller the block size the more accurately we can localize the tampering, this leads to another concept of sample wise authentication (Cox et al., 2008).

Even when the host signal gets degraded, restoration of portions of work that have been corrupted can be achieved through redundancy. One of the ways to introduce redundancy is to use the error correction bits in a cover work as a watermark. Considering the host signal as a collection of bits, the error correction codes (ECC) such as hamming, turbo, or Trellis codes are embedded as watermarks in this process. The size of the ECC determines the maximum number of bits that can be corrected in the host signal. Another technique to add redundancy is to self-embed a low-quality copy of the cover work as a robust watermark which can be extracted even when the original work gets tampered with.

2.3.2 Watermarking & Cryptography Analogy

Watermark embedding and cryptography can be considered analogous to each other. The watermark embedding and detection are similar to the encryption and decryption in cryptography. In the case of a simple symmetric cryptography we have an encryption function $E_k(.)$ that takes a clear text m and a key k to generate a cipher text, m_c, as

$$m_c = E_k(m) \tag{2.1}$$

The encryption protects the contents of the clear text during transmission. The receiver of the cipher text need to decrypt to get to the original message. After the decryption the text is clear again and is no longer protected. The decryption function $D_k()$ can be defined as

$$m = D_k(m_c) \tag{2.2}$$

similarly, in watermarking we have an embedding function, $\epsilon(.)$, that embeds a message m into original signal, C_o and generates a watermarked signal, C_w. This can be represented as

$$C_w = \epsilon_k(C_o, m) \tag{2.3}$$

The message is extracted out of the watermarked signal using a detection function $D()$. Many watermarking schemes use a key k to control the mapping between the message and the watermarked signal. In the case of an informed watermarking where the detector gets to know some aspects of the original signal, k can be assumed as a function of the original work, hence it can be assumed that in the case of an informed watermarking the detection can be as simple as associating a unique key with each work. Thus the detector of any watermark can be generalized as

$$m = D_k(C_w) \tag{2.4}$$

Though cryptography and watermarking share the same process steps, cryptography focuses on data disguise. It is a study of different methods to transmit data as a cipher or encrypted text so that the secrecy of the data is maintained until it reaches the recipient. Encryption can protect data during the transmission but once the data gets decrypted it loses protection. Watermarking, on the other hand, hides a message in plain text data. The message is embedded in the data and carried by the data that needs protection. In this process, a watermark or a tag/label is inserted into the data and it is extracted at a later stage to establish the source of the data (Mohanty, 2003). Watermarking embeds a message in a host signal in a way that its presence is not noticeable or in other words the secret message to be transmitted is hidden in the original signal. The branch of watermarking that deals with information hiding is called steganography.

The difference between steganography and cryptography is that though both are used to protect the information exchange, steganography does not give a clue that the data has been embedded with the watermark hence leaves attackers clueless as they do not see the need to decrypt the information. This makes steganography suitable for tasks where the encryption cannot be used. One such use case is copyright protection, where an additional hash appended to the message can be removed by the attacker where-as if a message is embedded using the steganography makes it hard to detect it and also remove it.

2.4 Watermark Design Requirements

Major applications of digital watermarking are the copyright protection of the data source, protecting the authenticity of data, and the unauthorized manipulation of the data. Based on the target application, requirements are set on the design of the watermark. Figure 2.1 depicts some major requirements of the watermarking

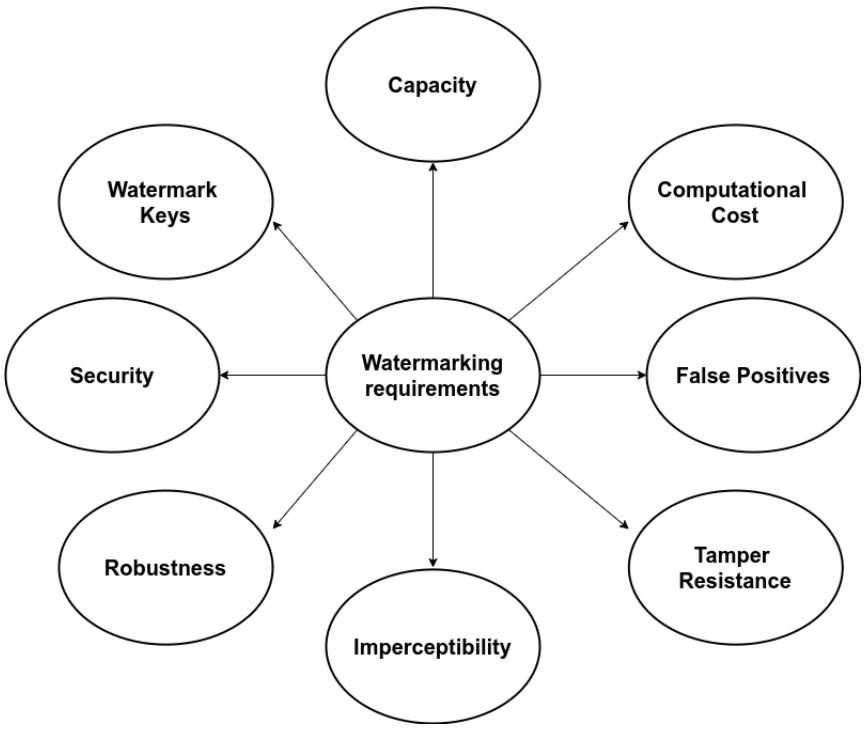


Figure 2.1: Watermarking requirements

techniques. The efficiency of the watermarking method applied in any application is measured based on these requirements. It is desired that any watermarking system provides a high data rate, low perceptibility or distortion, and yet high robustness. These three goals are mutually exclusive hence usually the performance of a watermark embedding system is characterized by the achievable trade-offs between these three goals (Brian and W, 2001). Here we discuss some of the general design requirements of the watermarking techniques

2.4.1 Imperceptibility

Imperceptibility or fidelity is an important feature of watermarking which is evaluated through the watermark induced distortion. Watermark-induced distortion should be so small that it should be extremely difficult to perceive it. For instance, in the case of the image watermarking, the watermarked image should appear the same as the original image. Both of them should be perceptually indistinguishable. To increase the imperceptibility of the watermarking process in images, some domain-specific steps like large singular values selection for watermark insertion are used. There are multiple ways in which the watermarked induced distortion is estimated. In the case of a mesh or 3D watermarking the change in the geometry is used as a method to evaluate imperceptibility. The most common methods to estimate geometric distortion being Hausdorff distance and signal to noise ratio.

2.4.1.1 Hausdorff Distance

Considering a geometric model in 3-dimensional Euclidean space represented by the 3-D point cloud. The non-watermarked data is, say M and the watermarked data is, say $M^{'}$ and $v, v^{'}$ where $v = (v_x, v_y, v_z)$ are the vertices of a mesh in the 3D geometric

representation of each model respectively, the Hausdorff distance is calculated as

$$D(M, M^{'}) = max(d_H(M, M^{'}), d_H(M^{'}, M) \tag{2.5}$$

$$d_H(M, M^{'}) = max_{v \in M}(d(v, M^{'})) \tag{2.6}$$

where, the $d(v, M^{'})$ is the minimum Euclidean distance between two geometric figures defined as

$$d(v, M^{'}) = min_{v^{'} \in M^{'}} |v - v^{'}| \tag{2.7}$$

2.4.1.2 Signal-to-Noise Ratio (SNR)

High signal to noise ration means better imperceptibility. The SNR between a watermarked signal model $M^{'}$ with vertices $x_i^{'}, y_{i'}^{'}, z_i^{'}; i \in \{1, \cdots, n\}$ and an original model M is computed as

$$SNR = 10 \log 10 \frac{\sum_{i=1}^{n}(x_i^2, y_i^2, z_i^2)}{\sum_{i}^{n}((x_i^{'} - x_i)^2, (y_i^{'} - y_i)^2, (z_i^{'} - z_i)^2)} \tag{2.8}$$

2.4.1.3 Root Mean Square Error (RMSE)

Another measure to evaluate the imperceptibility is RMSE. Lower RMSE between the watermarked and original models account for higher imperceptibility. The RMSE between the watermarked and non-watermarked models is calculated as

$$d_{rmse}(M, M^{'}) = \sqrt{\frac{1}{n} \sum_{i=1}^{n} |v_i^M - v_i^{M^{'}}|^2} \tag{2.9}$$

While the RMSE measures the geometric distance between the corresponding vertices in the watermarked and original models, it does not capture the subtle visual properties that human eye appreciates such as smoothness. These features are cap-

tured by more advanced Laplacian operators that take both topology and geometry into account. The geometric Laplacian of a vertex is given by

$$GL(v_i) = v_i - \frac{\sum_{j \in n(i)} l_{ij}^{-1} v_j}{\sum_{j \in n(i)} l_{ij}^{-1}} \tag{2.10}$$

where, n(i) is the set of indices of the neighbors of the vertex i, and l_{ij} is the geometric distance between vertices i and j

2.4.2 Tamper Resistance

Detection of tampering in the watermark can be used to check the authenticity of the original work. Any change to the watermark message can be considered as tampering with the original work. Hence the integrity of the data can be determined by checking the watermark. At the same time integrity of the data also determines the integrity of the embedded watermark. Again, it depends on what needs to be protected, the watermark, or the original data, and this decision is made based on the application. Based on the resistance to the tamper, the watermarks are categorized as fragile, semi-fragile, and robust. Bit Error Rate (BER) is used as a generic tamper resistance measure and it is used to measure the correctness of the method. A simple implementation of BER between watermarked model $M^{'}$ and the original model M is calculated as

$$BER(M, M^{'}) = 1 - \frac{1}{n_b} \sum_{n=1}^{n_b} \delta(m_i, m_i^{'}) \tag{2.11}$$

2.4.3 Robustness

A watermark is said to be robust in digital media if it can sustain any common signal processing manipulation operations implemented in the original work. For example, signal processing operations such as spatial filtering, lossy compression, image enhancements, cropping, and geometric transformations can eliminate the watermark

from the original image or a digital video and they could be used as attacks to beat the purpose of the watermark. Often the robustness cannot be seen as a single-dimensional value. A watermark that is robust against one process may be fragile against another. Designing a watermark that is robust against all the attacks can make it more complex to implement. Careful analysis of the application is necessary to understand the role of the watermark and then the robustness can be crafted based on the application needs. For example, in the case of broadcast monitoring applications, the watermark only needs to survive the transmission process. For a television broadcast, this means lossy compression, possibly analog to digital conversion. It need not be robust against rotation, spatial transformations, high pass filtering, or a variety of distortions that occur during the transmission and broadcast. Similarly, in other applications like clandestine communications where the cover media is transmitted digitally without compression the watermark need not be robust against spatial transformations. Also for applications such as sensor data integrity verification where the watermark is used for data integrity verification or to check whether the host data has been altered or not, the watermark need not be robust, in-fact it should be fragile or semi-fragile. Applications such as ownership verification, fingerprinting, copy control, etc need robustness since there is a high chance that hackers target the watermark and try to remove it. To measure the robustness, the most widely used approach is to extract the watermark from the distorted signal and evaluate the difference between the originally embedded watermark with the extracted one. The common method is to calculate the correlation coefficient as shown in the Equation (3.3).

2.4.4 Security

Security of a watermark becomes a key aspect in certain applications like copyright protection, data authentication, digital content tractability, and fingerprinting. To

ensure the security of the watermark several techniques such as logistic map based encryption, key-based encryption are used.

2.4.5 Capacity

Watermarking capacity is defined as the amount of information that can be inserted into the original work while satisfying the robustness and imperceptibility conditions for any given application. In other words, the capacity determines the information holding limitations of the watermarking scheme while maintaining other constraints such as imperceptibility and robustness. The watermark extraction depends on the density of the watermark embedding or the watermark packing capacity. For example, in the case of image watermarking, the extraction process is more successful when the number of bits embedded in the host image is high. High channel capacity increases the watermark detection probability and reduces the probability of false alarms but at the same time increases the image distortion. Depending on the application and its tolerance to the image distortion the embedding capacity needs to be defined. Methods such as Gaussian channel approximations and game theory-based approaches can be used to determine the capacity of a watermarking algorithm.

2.4.6 Computational Cost

The computational cost for embedding a watermark into a cover work and extracting it should minimal. The main criteria for estimating the computational cost is the time it takes to embed and extract the watermark, the complexity of the embedding and extracting algorithms, and their memory requirements. Usually, the more robust the watermark is the higher is the computational cost. Based on the application a good trade-off between the robustness and computational complexity needs to be maintained. The computational cost can also be viewed as time complexity of the

embedding or decoding algorithm. A common way to measure the time complexity is to use the rate of increase in data rate due to watermarking. For example, in a video frame if the original video stream has a bit rate R_x and the watermarked frame has a bit rate $\hat{R_x}$, assuming a constant time process to embed bits per fame the increase in bit rate can be computed as $\frac{\hat{R_x}-R_x}{R_x} \times 100$. This can be used as one of the measures to estimate the computational cost.

2.4.7 False Positive & False Negative

The false-positive rate is the characteristic used to identify watermarks in a work where there is no watermark embedded. This problem occurs when the embedded watermark is different from the extracted watermark. This measure also checks the efficacy of the watermark detector, especially when dealing with data integrity checks, the false positives should be minimal. The false positive rate can be computed as a ratio of the number of messages incorrectly identified as tampered to the total number of clean frames provided to the decoder

$$FP = \frac{N_{falsePositives}}{N_{clean}} \tag{2.12}$$

Similarly, the false-negative characteristic is used to identify if the watermark detector can properly detect the embedded watermark in the cover work. Again this measure brings some confidence in the watermarking system. In an ideal scenario this criteria should be again minimal or close to zero. This can be computed as the ratio of the number of cover work samples that are falsely identified as clean by the detector to the total number of tampered samples

$$FN = \frac{N_{falseNegatives}}{N_{tampered}} \tag{2.13}$$

2.4.8 Watermark Keys

Watermark keys constitute the generic term for secret information that determines the parameters of an embedding function. For example, in the case of image watermarking, the key could include host signal information like image coefficients or the embedding domain like spatial domain, or the transformations like DCT. Often, to ensure additional security in the embedding and extraction process of a watermark, a secret key is used. The accurate estimation of the watermark key by the receiver is important as it determines the degree of security of the watermarking system (Begum and Uddin, 2020). Like in cryptography, this key generation can be symmetric or asymmetric. In (Chopra et al., 2018) a method of using the XOR operation as a key to insert the watermark at a predefined location in a biometric signature is demonstrated. The side information used in informed watermarking can also be considered a type of key and in Chapter 4, a unique way to generate the side information for RADAR data integrity checks is discussed.

2.4.9 Watermark Reversibility

A watermark is considered reversible if the host signal can be reconstructed after the removal of the watermark. If a watermark is reversible, the extraction algorithm should be able to completely remove the watermark and reconstruct the host signal. This kind of watermarking could use side information or secret key in the process to generate the host signal or the original work and to extract the watermark.

2.5 Watermarking Techniques

There are three main characteristics of a watermark, stealth, robustness, and security. The embedded watermark should not be visible to the attacker, the data tampering should not affect the watermark and the legal owners of the data should

only be able to retrieve the embedded watermark. Giving more emphasis on any one of the attributes makes the other two weak, hence there is always a trade-off between these three important characteristics in any watermarking method (Sin-Joo Lee and Sung-Hwan Jung, 2001). Several watermarking techniques emphasize one characteristic vs another based on the end application needs. Watermarking can be divided into multiple categories in multiple ways. At a higher level, the classification can be done based on the working domain, document type, algorithm type, human perception-based, and application-based. Classification of the watermarking techniques based on these criteria is shown in Figure 2.2. Here we discuss the classification relevant to

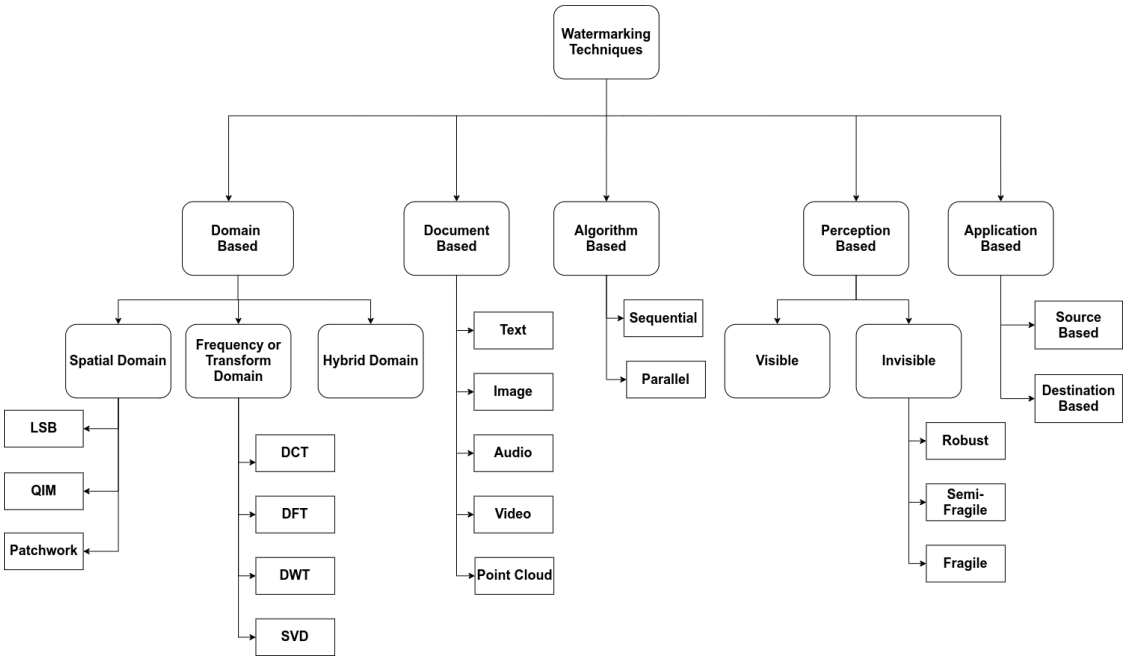


Figure 2.2: Watermarking techniques classification

our research. Based on the working domain, in particular to the digital media, the watermarking can be divided into spatial and frequency domain or a hybrid approach.

2.5.0.1 Spatial Domain Watermarking

When it comes to the spatial domain the host signal or data is embedded with a watermark in the spatial or time domain. The most common methods used in this

category are the Least Significant Bit (LSB), Intermediate Significant Bit (ISB), and QIM. These techniques are directly applied to the host data like the pixels in the case of images. The spatial domain watermarking techniques maintain an optimal balance between robustness, embedding induced distortion, and capacity. They are also easy to implement, computationally less complex, and have faster execution times. Because of their simplicity, any geometric manipulations to the host data can result in watermark removal. Another advantage of these methods is that they provide high data hiding capacity. Some of the most commonly used spatial domain watermarking techniques are

1. LSB
2. QIM

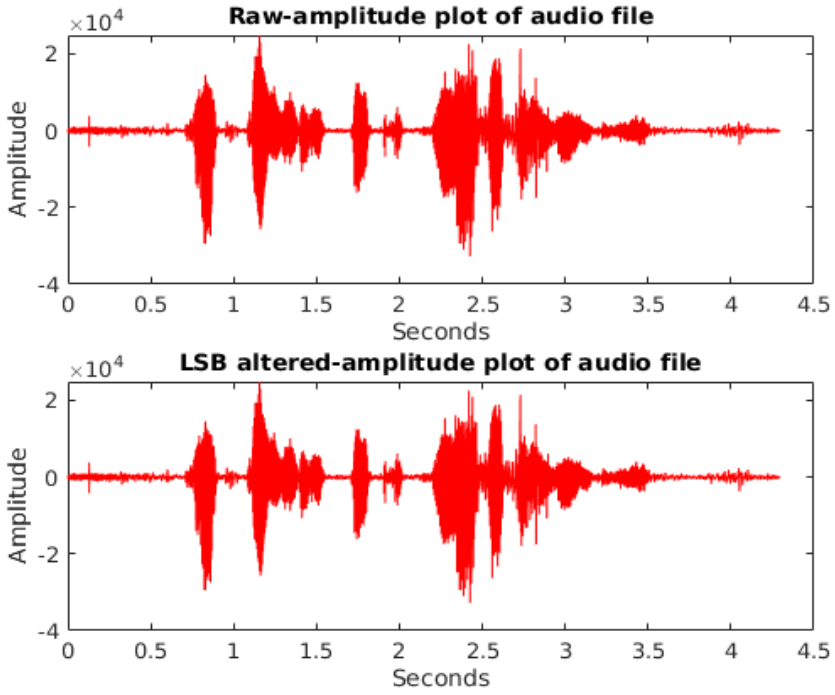


Figure 2.3: Amplitude plots of a raw and LSB modified audio sample

2.5.0.2 LSB

One of the simple data hiding techniques called the Least Significant Bit (LSB) encoding is explained here to provide a perspective on the ability of the data hiding techniques. In LSB or low-bit encoding, the least significant bit of each data sample is replaced by a binary value representing the hidden message. Thus for a signal of sampling rate 16 kHz, like an audio signal, the amount of data that can be hidden using this method is 16 Kbps. The embedded watermark can be easily extracted by reading the LSB of the received signal. The LSB in any data usually carries less critical information, hence, this method provides high imperceptibility. In Figure 2.3,

(a) Raw camera image

(b) LSB modified camera image

Figure 2.4: Representation of raw and LSB modified camera image

a time series plot of a single channel audio file is displayed. The audio file is a single-channel recording sampled at 16 kHz. It can be observed that there is no perceivable difference between the raw samples and the LSB embedded samples. With this simple LSB technique, another audio file is created in the region of acceptable fidelity in such a way that the overall quality of the host signal is not compromised, but at

the same time, a significant amount of hidden data i s transmitted over the channel. I n Figure 2.4, a 1242 ×375 ×3 i mage i s encoded by embedding i nverse of the least significant bit in each pixel. Again, it can be seen that there is no perceivable distortion, and yet a hidden message of size $1242 \times 375 \times 3$ bits are stored in the embedded image.

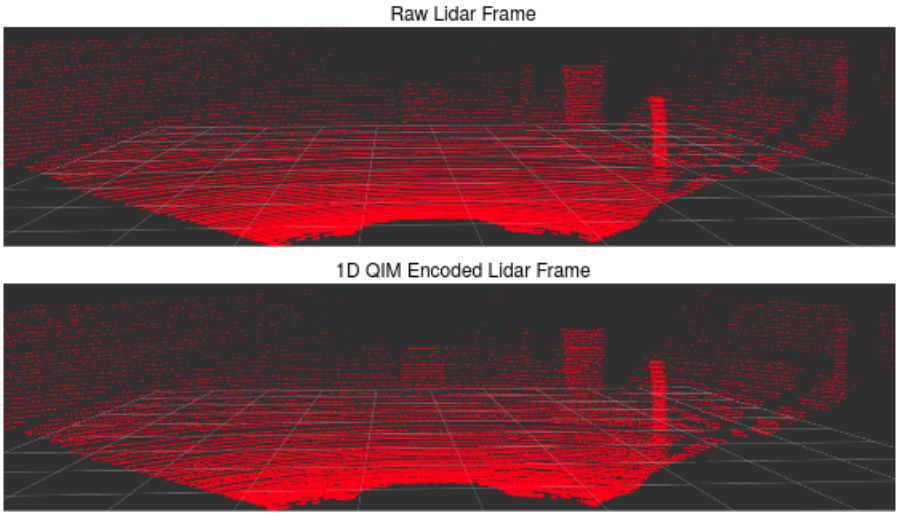


Figure 2.5: Representation of raw and QIM modified LiDAR image

2.5.0.3 1D QIM

Similarly, Figure 2.5 shows two frames of a 64-bit LiDAR from the KITTI vision benchmark dataset (Chen et al., 2017). One of the frames is embedded with a 1D-QIM data hiding method. The implementation is similar to the LSB embedding with a uniform scalar quantizer (Chen and Wornell, 2001). Further details on the Quantization Index Modulation are presented in chapter 3, section 3.4. The perturbation size chosen in this case is $\Delta = 3\ cm$, which means, each data sample is moved linearly by 3 cm after the quantization step. Again, we do not see any perceivable distortion in the frame quality after the data embedding. From these examples, it can be observed that irrespective of data dimensions, the data hiding techniques can be used to embed a watermark into the data and yet not distort the data to the extent

that it becomes useless. This embedded watermark can then be used to verify the integrity of the data when transmitted over a channel or a network.

2.5.1 Frequency Domain Watermarking

Spatial domain watermarking algorithms are applied directly to the host signal data and hence most of them are fragile. They can be easily manipulated which leads to the need to apply the watermark in the transformed domain or space such as frequency domain instead of the time or space domain. In these techniques, the host signal is transformed into a pre-defined space and the watermark is embedded into the transformed domain coefficients. At the receiving end, the inverse transform is applied to retrieve the watermark. These transform domain watermarking techniques are robust to different types of attacks like lossy compression and domain-specific signal processing like image sharpening, cropping, filtering, noise addition, etc. Some of the common domain transforming techniques are Discrete Cosine Transforms (DCT), Discrete Wavelet Transform (DWT), Discrete Fourier Transform(DFT), and Singular Value Decomposition (SVD).

2.5.2 Perception Based Watermarking

Human perception-based watermarking can be divided into visible and invisible watermark techniques. Visible watermarks are translucent whereas, invisible watermarks are highly imperceptible. They come under the steganography or data hiding techniques category and are more prevalent. The invisible watermarks can be further classified into three types

1. Fragile
2. Semi-fragile
3. Robust

Again, the choice of the watermarking stream depends on the end application goals and computational resources. For example, robust and semi-fragile watermarking techniques are generally used in copyright protection applications as those methods try to make watermark robust to any data changes. On the other hand, fragile watermarking gets distorted as the data gets tampered with and can be used to detect the check the integrity of data. Fragile watermarking is typically the choice of method in low power and low computational resource applications like Wireless Sensor Networks to implement data integrity checks on the data transfers from one node to another. Researchers proposed many fragile watermarking methods for resource-constrained WSNs.

2.5.2.1 Fragile Watermarking

In this form of watermarking the level of a watermark, security is at the lowest which means the embedded information can be easily removed. Any transformation the original work goes through will destroy the watermark. Most of the basic spatial domain watermarking techniques such as the Least-Significant-Bit fall under this category. LSB watermarking can be easily detected, extracted, and removed if the attacker knows the embedding technique. A similar technique to detect the location of the tampered image has been proposed in (Qin et al., 2017). The end goal of this watermarking is to make sure to detect the attacks on the integrity of the host signal or the original data.

2.5.2.2 Semi-fragile Watermarking

Semi-fragile watermarking techniques ensure that the watermarking endures the common application-specific modifications but they break if the host signal undergoes significant changes such as geometric transformations or forgery, etc. This watermarking technique is useful in applications such as media content transmissions where it is

possible to know the kind of transformations that are applied to the host signal while getting transmitted. The semi-fragile watermarking technique mentioned in (Kaur et al., 2019) demonstrates that for the digital media signals that need to undergo compression to save on bandwidth the semi-fragile watermarking protects against channel noise and frame drop attacks while providing detection and localization of more malicious attacks. Due to these characteristics, the semi-fragile spatial data hiding scheme QIM is used to address the sensor data integrity verification problem in our research.

2.5.2.3 Robust Watermarking

In the robust watermarking scheme, the watermark messages are securely embedded into the host signal with redundancy to make it difficult to erase the watermark. Copyright security is one of the primary applications of robust watermarks, as they allow the watermark embedded by the owner to remain intact. There are multiple ways to achieve robustness like redundancy or having multiple embeddings of the same watermark, self-referenced watermarks, and embedding at a low-bitrate. In (Parah et al., 2016), such a robust watermarking method for images that offers good protection against common data manipulation attacks like scaling, cropping, image rotation, and any combination of these attacks is discussed.

To conclude the introduction to watermarking, this brief overview of watermarking tells us that it is a multidisciplinary field (Moulin and Koetter, 2005). It consists of concepts and techniques from different areas such as signal processing, cryptography, coding theory, information theory, and computer science. For the problem we attempt to solve, which is sensor data integrity checks within the automotive domain, we chose to go with the fragile watermarking techniques and QIM method in particular. In the next chapter, we describe the generic watermarking models and explain how it transcends into scalar watermarking and lattice codes.

CHAPTER III

Watermarking & Data Models

3.1 Watermarking Model

In this chapter, we provide a generic overview of the watermarking process starting with the sensor data modeling. The three basic components of any watermarking scheme are watermark embedding, detection, and extraction. These three major processes are implemented as functions that act on the corresponding input data. Here we try to understand the geometrical meaning of these watermarking steps as-well-as their analogy to the traditional communication models by getting into the details of each step.

3.1.1 Data Model

To define the data model for multi-dimensional data, we use the notation of an object and its features (Dzemyda et al., 2013). The object can be an instance or a sample of the data and the features are the properties or the attributes that determine the dimensions. Data samples that can be described by a set of features can be considered as a data-set $X = \{x_1, x_2, \cdots, x_m\}$ where, m is the length of the data-set. If we use this notation, a particular sample x_i of the data-set contains a combination of values of all the features, represented as $x_i = \{x_{i1}, x_{i2}, \cdots, x_{in}\}$, where n is the number of features. The length of the feature set representing a sample can be

considered as the number of dimensions and the data-set samples are represented as points in n-dimensional feature space $\in R^n$.

The inputs to a watermark embedding function are the original unwatermarked data C_o, a message to be embedded m, and an optional key k. The embedder is a deterministic function that maps these inputs to produce a watermarked work C_w as an output. In watermarking terminology, the cover work/original work/the host signal C_o is the media content into which the watermark is embedded. Without the loss of generality, C_o can be considered as a set of points in n-dimensional media space. If we use the notation of sample and feature set explained above, the cover work with n different features can be represented as a set of m vectors, $C_o \in \{c_0, \cdots, c_m\}$ in an n-dimensional media space. Here, the media space can be considered as a subset of R^n similar to the Euclidean space. The dimensionality of the signal depends on the application and the selected sample space.

Let us take a look at the notations used to represent different digital media content such as grayscale and color images, digital audio, video, and 3D point cloud geometry. In the case of digital images, the dimensionality or the feature set depends on the notation of the sample space. An image can be represented in a pixel or a frame sample space. In a pixel sample space, for a grayscale image, the set of pixels can assume a flat structure. This can be represented by a 1-dimensional tuple of m number of pixels per image. An example of such representation is a lexicographic scan image. In a frame sample space the position of pixels in (x, y) plane can be considered as two features, hence they are represented in a two-dimensional Euclidean space E^2, hence n becomes two. Similarly, a color image with three color planes (RGB) and m pixels per plane can be represented in three-dimensional feature space as $[x, y, channel]$, hence $n = 3$. These sample space notations for grayscale and color images are depicted in Figure 3.1. Also, in the digital media space, the content value is quantized and bounded. This leads to a finite set of possible works in the media space that can be

arranged in rectilinear lattices. If we consider the example of a grayscale image the pixel values can be quantized within the integer range of values 0 to 255. Since the data points are quantized with a specific step size form a lattice, the values between the lattice points or that fall outside the bounds cannot be considered as part of the original work. However, by adjusting the quantization step-size and making the bounds large enough the media space can be considered continuous.

The above notation can also be extended to continuous or temporal content such as audio & video. A digitized audio signal is a set of samples collected over time. These samples can be divided into fixed-length segments that decide the length of the signal m. The dimensionality of each segment is the time at which it is collected hence, for an audio signal, $n = 1$. In the case of video, n would be the product of several frames in a fixed segment, the number of pixels per frame, and three channels (RGB planes in the color video). In the case of spatial point clouds such as the LiDAR point cloud, each data point in the point cloud can be visualized by its position coordinates $[x, y, z]$, hence here $n = 3$. In each dimension the data points are represented as p, q, r-dimensional vectors, and the quantized data is represented as linear, square, and cubic lattices respectively. In Figure 3.1, the media space representation of LiDAR data in an orthogonal coordinate system is shown. There are a lot of n-dimensional spaces but in the context of this research, for the most part, we restrict it to spaces similar to Euclidean space of n-dimensions with orthogonal global coordinates. Under this assumption, a point in this n-dimensional plane can be considered as a vector from the origin to the designated set of coordinate values. In Euclidean space, any n-dimensional vector can be represented as a tuple of points in an n-dimensional space R^n as $x = (x_1, x_2, x_3, \cdots, x_n)$, where $x_i \in R$. With this notation, a line, plane, and three-dimensional space can be represented by setting n = 1,2, or 3 respectively. A point in the three-dimensional Euclidean space can be considered as a set of triples $(x, y, z) \in E^3$ (Hanson, 1994). When we are dealing with spatial data such as the

point sampled geometry of a 3D model or LiDAR point cloud with spatial position information, etc. the Euclidean space and media space can be used interchangeably.

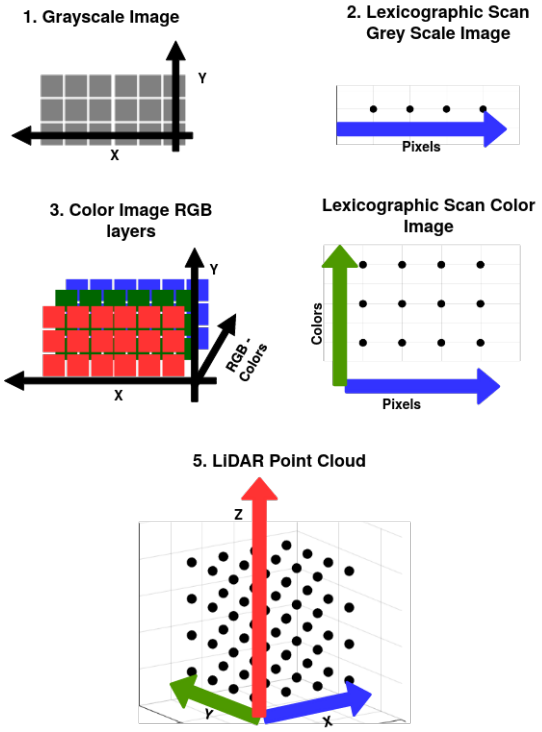


Figure 3.1: Digital content representation

3.2 Geometric and Statistical Models

To view the watermarking process in a geometrical representation we consider a high-dimensional space in which each point corresponds to one watermarked work. Here the watermarked work is a type of watermark applied to the data in the media space. This high dimensional space is called media space during the the embedding phase. In the detection phase, it is referred to as marking space and represents the watermarked works. The marking space can be considered as the projection or transformation applied on the original work. These transformations such as frequency transform, filtering, block averaging, geometric or temporal registration, etc. are applied as a part of the watermark embedding and extraction processes.

Geometrically, the watermarking system can be viewed as various regions of probability distributions in the media and marking space such as the distribution of unwatermarked works, detection region, embedding region, and the region of acceptable fidelity. In the watermarking process, though the embedded function itself is deterministic the randomness in the embedder's output comes from the fact that the original works are drawn randomly from the distribution of unwatermarked data. If multiple unwatermarked data distributions are mapped into the watermarked work C_w, the probability of C_w which is known as the embedding distribution is just the sum of the probabilities of all the input distributions.

Distribution of *un-watermarked works* becomes useful in estimating the false positives in watermark detection. It estimates the likelihood of watermark embedding in a media space. It is important to have a priori distribution of the content we want the system to process since the watermarks are embedded in certain regions of the media space. Like in an audio file the probability of embedding a watermark in the music is higher than in the static. To represent this probability distribution over the media space (lattice of points that represent digital work) or the probability density function over all the points in the media space an elliptical Gaussian is considered as a simple statistical model that fits this distribution. To understand the region of *acceptable fidelity* consider an original work or the host signal in media space C_o such as an image. Just by altering a single pixel unit of brightness a new image can be created which represents a new vector in the media space and yet perceptually indistinguishable from the original image. If we can imagine a region around C_o which constitutes the set of perceptually indistinguishable images from C_o, this region in the media space is called the region of acceptable fidelity. As explained in chapter 2, there are multiple ways to estimate the perceptual distance but the most common method used to measure the perceptual distance metric is Mean Square Error (MSE)

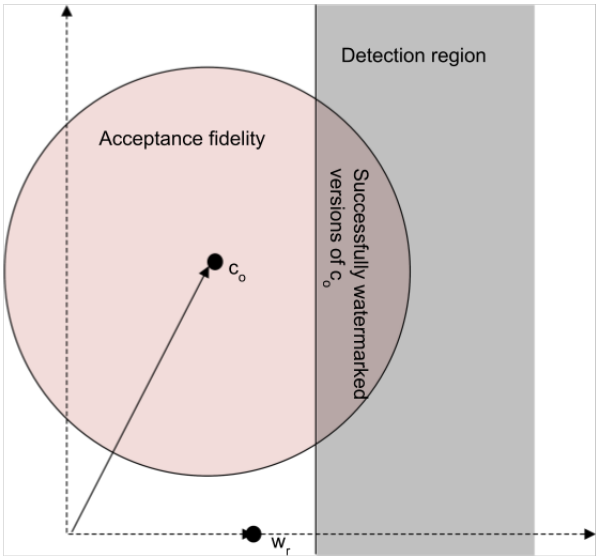


Figure 3.2: Geometrical model representation

given as

$$D_{mse}(C_1, C_2) = \frac{1}{N}\sum_{i=1}^{N}(c_1[i] - c_2[i])^2 \tag{3.1}$$

where C_1 and C_2 are N-dimensional vectors in the media space. If we set a limit of τ_{mse} on this function, the region of acceptable fidelity becomes an N-dimensional ball of radius $\sqrt{N\tau_{mse}}$. In some practical cases, the asymmetric distance between an original work and its modified version is calculated based on the reciprocal of the signal to noise ratio (SNR)

$$D_{mse}(C_1, C_2) = \frac{\sum_{i=1}^{N}(c_1[i] - c_2[i])^2}{\sum_{i=1}^{N} c_1[i]^2} \tag{3.2}$$

The *distortion distribution* tells us how likely the watermarked signal C_w is to be distorted during normal usage or it gives the likelihood of obtaining a distorted work C_{wn}, given the watermarked work C_w. For example, in normal digital media operations, the content distortion is caused by signal processing functions like filtering, noise reduction, temporal or geometric distortion and lossy compressions. This distor-

tion can be modeled as an additive Gaussian noise channel thus making it dependent on the content itself.

The *detection region* for a given message, m and a watermark key k is the set of works in the marked space that will be decoded by the detector as containing the message. The linear correlation measure between two vectors is the average of the inner product between vectors. This method is used in communications to check the presence of a transmitted signal in the received signal. The linear correlation coefficient is computed and checked against a threshold value to check the presence of the transmit signal.

Similarly, in watermarking, linear correlation may be used as a measure to check the presence of a watermark. This can be computed as the inner dot product or the product of the received watermark signal C_w and the reference watermark W_r divided by the length of the signal N

$$z_{lc}(C_w, W_r) = C_w \cdot W_r / N \tag{3.3}$$

If we want to use the correlation coefficient, we need to subtract the mean values from the original vectors before performing the normalized correlation.

Geometrically, Equation (3.3) can be viewed as an orthogonal projection of C_w into the constant length vector W_r. The values of C_w that are greater than a predefined threshold τ_{lc} fall on one side of the plane perpendicular to W_r. This results in a detection region comprising of the points belonging to one message and that leads to the embedded message recovery. If we look at a simple case of binary message $m \in \{0, 1\}$, one side of the plane perpendicular to W_r as shown in Figure 3.2, corresponds to points represented by $m = 1$ or τ_{lc} and the other represent $m = 0$ or $-\tau_{lc}$.

A watermark embedding process is considered successful if the embedding results in a work that falls in the intersection of the regions of acceptable fidelity and the

detection as shown in Figure 3.2. Here a 2D media space is represented by the c_o and reference watermark by w_r vector. The circular area of acceptable fidelity intersects the detection region to the right. The distance of the detection region from the origin is determined by τ_{lc}. This concept can be extended to higher dimensions where the detection region becomes a hyperplane orthogonal to the reference watermark plane w_r. If we pick high dimension vectors from Gaussian distribution, they tend to be perpendicular to a given reference watermark plane. Thus, if we add Gaussian white noise to the host signal vectors, the corrupted vectors tend to be parallel to the detection region plane and hence easily distinguishable.

3.3 Communication Model

In this section, we look at watermarking as a communications problem. Thinking of the watermarking method in terms of a digital communication channel enhances our understanding. A generic watermarking system consists of two primary stages, the embedder, and the decoder. The embedder combines a message m with the host signal and generates a watermarked signal. As this watermarked signal gets to the decoder additional distortion that could be both natural and malicious may get added to the signal. The decoder extracts the embedded message from the resulting signal using the auxiliary information. In a typical communication problem, there exists a sender and receiver pair and a medium of communication between them known as a transmission channel. If we look at the analogy between a watermarking process and the communications systems, the sender in a communication system is analogous to the embedder and the decoder is analogous to the receiver and the distortion can be represented by the channel noise.

The sender has the signal to send to the receiver called as host signal in the media space $C_o \in R^n$. This signal is allowed to be modified within a certain limit and the resulting signal is the watermarked signal $C_w[m]$, where $m \in M$ is the message

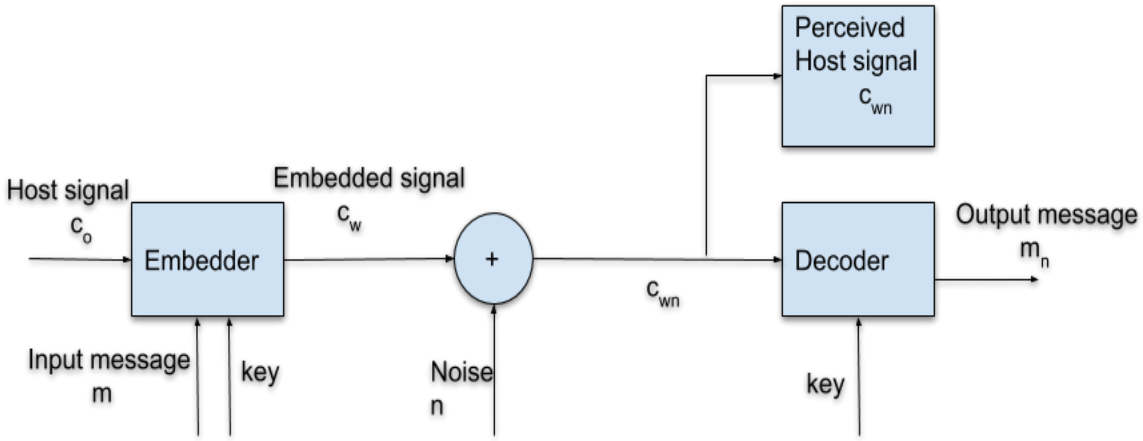


Figure 3.3: Watermarking as communication model

to be embedded. In informed watermarking the embedded watermark depends on a message mask based on the key k, the decoder has access to or the watermark is derived out of the original work itself and the decoder is aware of the watermark generation algorithm. The embedding process can be represented as

$$C_w[m] = f(C_o, m, k) \tag{3.4}$$

a function of the original work C_o, message to be embedded m and the side information key k.

The category of a watermarking model that fits the sensor data integrity verification problem we are trying to solve in this research is the side information based blind decoding. In this model, the detector is not aware of the original signal, but the auxiliary information such as the encoding message sequence or the key can be used to make integrity verification more accurate (Cox et al., 2008). The applications presented in the subsequent chapters are based on this model. In Figure 3.3, blind watermark detection is depicted as communication of message m and the host signal C_o over a communication channel. This embedding process induces distortion that is constrained by the enforced distortion limits based on the end application. This embedding induced distortion metric, d_e, between the original work and the

watermarked work, represented as $d_e(C_o, C_w[m])$, should be less than a threshold p_e. The watermarked message undergoes distortion and degradation in the transmission channel t_c that degrades the watermarked signal $C_w[m]$ to $C_{wn}[m]$ that is received at the receiver end. The channel degradation considered here can include both the required signal modifications such as a lossy compression as well as the malicious attacks on the signal such as unwanted modifications. A completely distorted signal will be of no value to the receiver, hence we can put some constraints on the channel distortion. The channel t_c is constrained such that the channel induced distortion $d_c(C_w[m], C_{wn}[m]) < p_c$. The decoder receives $c_{wn}[m]$ along with the side information k. With this information, the decoder estimates the embedded message m_n, of the original embedded message m.

In communication systems, the channel efficiency is assessed by measuring how intact the received signal is. Similarly, in watermarking, the decoder's ability to estimate the embedded message accurately is used to measure the performance of the watermarking scheme. One of the metrics to measure this performance is transmission error rate. This metric is defined as the probability that the embedded message m is different from the decoded message m_n. Also, the rate R of the watermarking algorithm is another metric used to measure the performance and it is defined as $R = \frac{\log_2 |M|}{N}$ where $|M|$ is the length of the message symbols and N is the number of samples in the cover work C_o. In other words, the rate defines the number of message bits per symbol in the cover work.

If we model the original work C_o as a random sequence of real i.i.d values $\mathbf{s} = \{s_1, s_2, s_3, \cdots, s_n\}$ over a media space $\mathbf{S}$. Given a message m, uniformly drawn from a message set M, the sender produces a watermark $\mathbf{x} = \{x_1, x_2, \cdots, x_n\}$, such that the embedding induced distortion $d(\mathbf{s},\mathbf{x}) < p_e$. The distortion between $\mathbf{s}$ and $\mathbf{x}$ can

be measured using the common MSE function represented as

$$d(\mathbf{s},\mathbf{x}) = \frac{1}{n}\sum_{i=0}^{n}(s_i - x_i)^2 \tag{3.5}$$

The transmission channel can be defined based on the choice of noise function applied. In this study, we consider an additive noise channel where the signals are modified by the addition of the noise. The noise can be drawn from a distribution independent of the host signal distribution. In the simplest form, this noise can be modeled as a Gaussian channel where the noise elements are drawn independently from a normal distribution with zero mean and σ_n^2 variance. The transmission channel t_c, can be modeled as a AWGN that modifies the watermarked work $\mathbf{x}$, with noise samples in $\mathbf{n} = \{v_1, v_2, \cdots, v_n\}$. The resulting signal $\mathbf{y}$ can be represented as

$$\mathbf{y} = \mathbf{x} + \mathbf{n} \tag{3.6}$$

Decoder receives this signal and estimates the embedded message m. We assume that the sender and receiver are aware of the channel power p_c and not unaware of the noise values v_i. In this side information model, the sender and receiver share the knowledge of the signal power Q, the embedding power p_e and the message set M along with any auxiliary information but the receiver is not aware of the original work, $\mathbf{S}$.

With these assumptions, if we consider the case of single cover work, the formulation is equivalent to the classical problem of communication over an AWGN Gaussian channel and the transmission rate R of such system can be given by Equation (3.7) based on Shannon's theorem of the capacity of an AWGN channel

$$R = \frac{1}{2}\log\left(1 + \frac{P_e}{P_c}\right) \tag{3.7}$$

If we consider multi-message works, in a more general case, the encoder needs to transmit a separate watermark signal s_w within the embedding power constraint p_e, for every host signal s and each message m. Here, different sets of messages are grouped by code words. The receiver needs to retrieve the message from the signal degraded during transmission. For the receiver to be able to unambiguously decide the received message, the code words for each message needs to be at a sufficient distance from each other, and at the same time to achieve the embedding power constraint they need to be densely packed. To fulfill this trade-off the codewords need to be optimally selected and cannot be randomly picked from a set of uncorrelated Gaussian random variables. To achieve this we need to part away from the one to one correspondence between a message and a code word and get to a notion where one to many mappings between a message and a code word exist such that there is a choice to pick the codeword that better fits the host signal for any given message m. This notation is called a dirty paper code.

In a dirty paper code technique, the code words that represent a message are picked from the host signal to reduce the embedding distortion. The dirty paper code scheme can be explained as, for each message m a code C_m is selected such that the watermarked output $\mathbf{x} \in C_m$ is pretty close to the original signal $\mathbf{s}$. At the receiver end from the degraded signal $\mathbf{y}$ as shown in Equation (3.6), the receiver finds the code word $\mathbf{x}^{'}$ in the set $C = \cup_m C_m$ which is an union of all the sub code books and finds the index of the sub-code book to which $\mathbf{x}^{'}$ belongs to.

If we think that each codeword is generated by a high-dimensional quantizer for a given message m, all signals in $\mathbf{s}$ that are close to $\mathbf{x} \in C_m$ are represented by $\mathbf{x}$. This can be further explained using a simple example. Consider $C = \{C_0, C_1\}$ correspond to the union of the even and odd integer codebooks and a binary message set $M = \{0, 1\}$. When the sender wants to send a message embedding 0 or 1, an even or odd integer nearest in value to the unwatermarked signal $\mathbf{s}$ is transmitted and the

receiver decodes the embedded message by picking the closest integer in C. If this integer is even then the embedded message should be 0 and 1 otherwise.

Now that we looked at the watermarking process as a communication problem and estimated the core features such as data hiding rate and the information embedding capacity, we can take a closer look at the concept of generating the watermark. In the case of blind watermarking, the generated watermark is assumed to be independent of the original work or in other words, the correlation between the original work and the watermark is very minimal. In the majority of practical watermarking applications like the sensor data integrity verification, the unwatermarked data is known to the embedder, this is called informed encoding. So the watermark w_a, in this kind of embedding is dependent on the original data C_o or the data is provided as an additional input to the encoder. Given this scenario, the watermark generators take advantage of this knowledge and generate watermarks that have a low correlation with the original data. Usually, to make sure that the linear correlation between the original data and the message pattern is minimal, the watermark generator first computes the linear correlation between the original data and the watermark and adjusts the amplitude of the added pattern to compensate or reduce the correlation. In the data fidelity check applications, where the watermark is considered as a noise, the document to noise ratio is kept high. In the case of applications that require watermark effectiveness or robustness, where the original data is considered as the noise, the watermark to noise ratio is kept high. These adjustments are done while generating a watermark and the optimum watermark is picked for a given application.

Also, the host-signal interference occurs from the systems that do not necessarily let the embedder exploit the know-how of the host signal. In these cases, the host signal itself acts as a noise and makes it difficult to decode the watermark and extract the embedded message. One of the reasons for selecting the Quantization Index Modulation (QIM) for our research is because it offers host signal interference rejection

as it can exploit the knowledge of the host signal at the encoding phase. In QIM, even in the absence of any perturbations, there will be quantizers that will let us uniquely determine the embedded message m, this non-intersection property of the quantizers is the reason behind the host-signal interference rejection property of the QIM method (Brian and W, 2001). The watermark interference with cover work can be examined using the perceptual models and the watermark magnitude w_a can be adjusted to reduce the interference.

Another way to look at the watermarking process is as a multiplexed communication. The embedder combines the watermark and data into a single signal C_w. Original data is considered as a second message to be transmitted along with the watermark message in the same signal. After the signal passes through the transmission channel the watermark detector and the system that perceives or processes the original data both receive the same data C_{wn}. Here, two different receivers decode two different messages C_o and m. The system that does data processing on the watermarked data should perceive or get data that is close to the original data. Hence, the watermark induced distortion should be minimal for this system. On the other hand, the watermark detection system should be able to obtain the watermark without interference from the original data, hence, in this case the watermark fidelity is an important criteria. The applications listed in upcoming chapters use this kind of strategy to parallelize the watermark extraction and the signal decoding processes.

3.4 Scalar Watermarking

Watermarking methods that require each sample of the host signal to be separately quantized or the methods that put restrictions on the codebook separation are known as scalar watermarking methods. The capacity of an AWGN channel represented by 3.7 can be ideally achieved by watermarking schemes like ideal Costa schemes (ICS). Without going into details, we can assume that the ICS achieves the

maximum possible transmission rate because of the inclusion of codebooks that are spread across high dimensional space and due to distortion compensation. The comparison between ICS and other practical scalar schemes such as the spread spectrum, scalar costa scheme which can also be considered as distortion compensated QIM, and the Dither modulation shows that the ICS outperforms all the scalar methods as long as they are one dimensional but still these methods are effective ways to hide information if data rate and capacity are not primary constraints of the application, scalar watermarking methods are easy to implement or computationally less intensive (Cox et al., 2008). The most popular scalar watermarking methods are LSB, QIM, distortion compensated QIM, and scalar Costa scheme.

3.4.1 Binning Schemes - LSB

The basic scalar watermarking method is the LSB watermarking. As mentioned in chapter 2, this method modifies the least significant bit in the host signal to embed a watermark. If we take an example of an image as the host signal, where each pixel is represented by a byte or 8-bit value the pixels are modified to carry 1 bit of information by changing the least significant bit (LSB) of the value. In a more generic case, the LSB can be viewed as a type of binning scheme. Consider a sequence $\mathbf{S} = \{0, 1, \cdots, 2^b - 1\}$ partitioned into two bins such as even and odd integers. $S_e = \{0, 2, \cdots, 2^b - 2\}$ and $S_o = \{1, 3, \cdots, 2^b - 1\}$, here S_e and S_o can be considered as two bins. Let $\mathbf{S}$ be a binary sequence of length N. Let X be the marked new sequence generated by embedding the binary message $m = \{m_1, m_2, \cdots, m_n\}$ into $\mathbf{S}$ and the embedding distortion constraint that X should satisfy is $|X - s_i \leq 1|$ for $1 < i < N$. The LSB function can be implemented as

$$x_i = m_i + 2\lfloor s_i/2 \rfloor, \quad 1 \leq i \leq N \tag{3.8}$$

we choose value of x based on the input message m

$$x_i = \begin{cases} x_i \in S_e & : \text{if } m_i = 0 \\ x_i \in S_o & : \text{if } m_i = 1 \end{cases} \tag{3.9}$$

We can interpret the LSB as a binning scheme in this regard where the S_e or S_o are the two bins from which the value of x_i can be chosen based on the value of m_i.

3.4.2 Quantization Index Modulation - QIM

The QIM and its variants of scalar watermarking methods were introduced by Chen & Wornell in 1999. These methods embed signal-dependent watermarks using the quantization technique and fall under the semi-fragile spatial watermarking techniques category (Brian and W, 2001), (Chen and Wornell, 1998), (Chen and Wornell, 2001). QIM can be viewed as an extension of the binning scheme explained above. Let's say we have a set of L messages to be embedded into a host signal of length L in L-dimensional space, QIM defines a unique set of L reconstruction points each mapped to a quantizer in a set $Q = \{Q_1, Q_2, \cdots, Q_L\}$ (Joachim and Bernd, 2002). The watermark embedding and decoding mechanism using the QIM method is fairly simple. To embed a watermark message m, the host signal x is quantized using the quantizer Q_m to obtain a watermarked signal s_w that is within a distortion constraint d_e. The watermark Q_m is chosen by modulating the index of the quantizer Q with the message to pick a quantizer from the set. While decoding, the receiver quantizes the signal received s_r by the union of all the quantizers used in embedding $Q = \{Q_1, Q_2, \cdots, Q_L\}$ and the index of the reconstruction point with the quantized value nearest to the received signal is identified and the corresponding watermark information $\hat{m}$ is extracted.

The basic idea can be explained by looking at a simple problem of embedding one bit in a real-valued sample. Consider a binary message $m \in \{0, 1\}$ is a one-bit

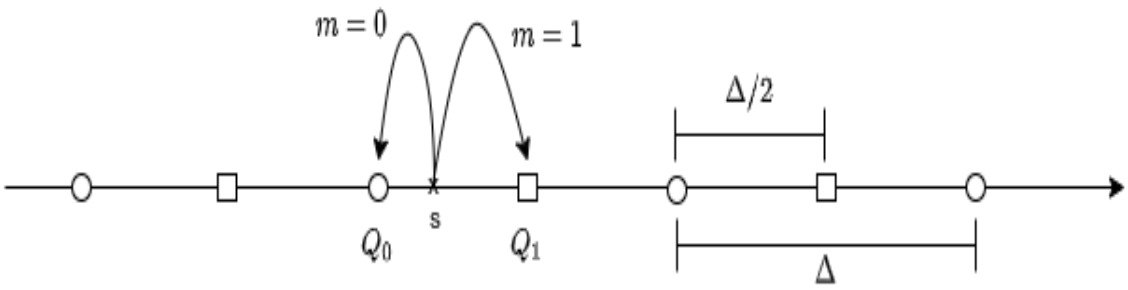


Figure 3.4: 1D QIM scheme

message, and $s \in R$ is one sample without any key k. A scalar uniform quantizer with step size Δ is defined as

$$Q_\Delta(s) = \lfloor s/\Delta \rfloor \Delta \tag{3.10}$$

Figure 3.4 depicts the 1D QIM where two uniform quantizer sets represented by circles and squares are shown on a real line. The quantizer set Q_0 is defined by uniform quantizer as shown in Equation (3.10), with a step-size Δ. The second set of quantizers Q_1 is represented by squares with an offset of $\Delta/2$. Thus any point falling in between the two quantizers is assigned a circle or a square based on the embedded message value $m \in \{0, 1\}$. At the receiver end the value is error-free as long as the distortion added is $\leq \Delta/4$. The scalar quantizatizer in Equation (3.10) can be extended to a dither quantizer in Equation (3.11) by adding dither d_i values

$$Q_i(s) = Q_i(s - d_i) + d_i, \qquad i = 0, 1 \tag{3.11}$$

for a simple dither quantizer, we can select fixed dither values like

$$d_i = \begin{cases} \frac{-\Delta}{2}, & i = 0 \\ \frac{\Delta}{2}, & i = 1 \end{cases} \tag{3.12}$$

To increase the randomness in the quantized values, the dither values can also be chosen from a pseudo-random uniform distribution over a given range. Given the

above example, if d_0 is chosen randomly over a distribution $[-\Delta/2, \Delta/2]$ then the dither d_1 can be chosen based on the d_0 value as follows.

$$d_1 = \begin{cases} d_0 + \frac{\Delta}{2}; & if,\ d_0 < 0 \\ d_0 - \frac{\Delta}{2}; & if,\ d_0 > 0 \end{cases} \tag{3.13}$$

where the dithers d_0 and d_1 embed bits 0 and 1 respectively. The watermarked signal for a signal of length L is given by

$$y_n = Q_\Delta(x_n, d_m) = Q_\Delta(x_n - d_m) + d_m,\ \text{where},\ m \in [0, 1], \qquad n = 1, 2, \cdots, L \tag{3.14}$$

If the quantization errors are uniformly distributed over $[-\Delta/2, \Delta/2]$, and the quantization cells are sufficiently small, then the mean squared distortion due to embedding is given by

$$d_e = \Delta^2/12. \tag{3.15}$$

This watermarked signal when transmitted over a noisy channel or gets corrupted by attacker results in a noisy signal as per Equation (3.6). In the QIM process, at the receiving end, the decoder used is a minimum distance decoder. It finds the quantizer point closest to the y and outputs the estimated message. In our binary message $m \in \{0, 1\}$ example, the decoder can be represented using the following equation.

$$\hat{m} = \underset{m \in \{0,1\}}{argmin} ||y - Q_\Delta(y, d_m)|| \tag{3.16}$$

In the case where the message is spread over N samples $\{x_1, x_2, \cdots, x_N\}$, which is usually the case in dither modulation, to improve the robustness, the delta between the received signal and the quantized value can be summed across N samples as

$$\hat{m} = \underset{m \in \{0,1\}}{argmin} \sum_{i=1}^{N} ||y_i - Q_\Delta(y_i, d_m)|| \tag{3.17}$$

Scalar watermarking schemes discussed above are one-dimensional. Optimal watermarking rates for these schemes can be achieved if the size of the signal is sufficiently large. To increase the watermarking capacity, for a given watermark to noise ratio, we need to increase the message set M in an M-ary scalar watermarking scheme to a larger value and resort to higher dimension vector quantizers. In high dimensions, an M-ary scalar watermarking can be generalized by defining the super-set codebook C consisting of all the multiples of step-size Δ as

$$C = \{k\Delta | k \in Z\} \tag{3.18}$$

which is the union of the sub-code books, C_m defined as

$$C_m = \{(m + kM)\Delta | k \in Z\}; \quad m = \{0, \cdots, M\} \tag{3.19}$$

The closest value x_m in the sub-codebook C_m is computed as

$$x_m = Q_{M\Delta, m/M}(s) \tag{3.20}$$

where the quantizer Q_Δ is defined by Equation (3.10). The embedding induced distortion of Equation (3.15) becomes $P_e = M^2\Delta^2/12$, for a uniform distribution of Δ values over a small quantization interval. Again, the embedded message is decoded from the received symbol by searching the point in the set C, with value closest to y, and determining the sub-code book to which this point belongs to.

$$m = \lfloor y/\Delta \rfloor \bmod (M) \tag{3.21}$$

In the case of the distortion compensated QIM, we bring a scaling factor α to control the step-size and the host signal values are quantized using a step-size Δ/α. By controlling the α we can control the embedding distortion hence it's also called the distortion compensation parameter.

$$s_m = Q_{M\Delta,m/M}(\alpha s) + (1-\alpha)s \tag{3.22}$$

To implement the higher dimension vector quantizers, we need to generate codebooks in high dimensions, and doing this randomly is computationally intensive as it involves searching for the best match code words in unstructured codebooks. For most of the practical applications where the signal lengths are constrained and where the generalized multidimensional codebooks are not available, we still want our codebooks to be separable or need each sample of the host signal to be separately quantized with an optimal transmission rate. A rather simple approach to achieve this would be to go with structured codebooks in higher dimensions such as, lattice codes, which are proved to provide better performance than 1D scalar codes.

3.5 Lattice Codes

A lattice is defined as an infinite set of points arranged in a repeated structure filling the space and the geometry that can be generated by repeated translations of the single starting point (Hanson, 1994). A space lattice is defined by a set of repeated points in m-dimensional real Euclidean space R^m that fill the space. The set can be generated from k linearly independent vectors $\{a_1, \cdots, a_k\}$ called generating vectors of the lattice. The lattice points are formed by the linear combinations of these generating vectors

$$R = n_1 a_1 + n_2 a_2 \cdots n_k a_k \tag{3.23}$$

where $n_i \in \{Z\}$ and k represents the lattice Λ dimensions. The core basis unit of a lattice that gets repeated is called a unit cell. In two dimensional lattice the unit cell is an area and in three dimensions its a volume. Example of a two dimensional lattice, is a Bravais lattice represented as $R = (n_1 d, n_2 d)$. It is a simple cubic lattice with lattice spacing d. The generator vectors for simple cubic lattice are $a_1 = (d_1, 0)$ and $a_2 = (0, d_2)$. There are also different other forms of 2D Bravais lattices like rectangular lattice and hexagonal lattice with generator vectors $a_1 = (d, 0)$ and $a_2 = (d/2, \sqrt{3}d/2)$ as shown in Figure 3.5 (Cleland, 2003). In three dimensions, again,

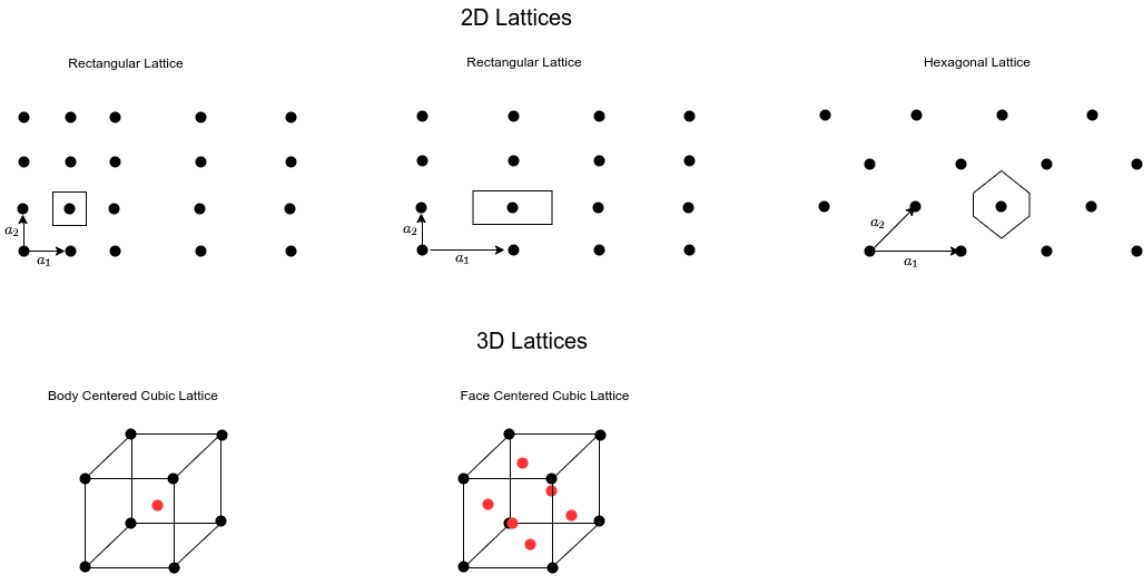


Figure 3.5: Simple 1D and 2D lattices

a simple representation of a 3D lattice is a simple cubic. The simple cubic lattice consists of points spaced along the three coordinate axes $R = (ld, md, nd)$ for $l, m, n \in \{Z\}$ and d being the lattice spacing. This geometry can be further extended to face-centered and body-centered lattices as shown in Figure 3.5.

The root lattices are the n-dimensional latices denoted by $A_n (n \geq 1)$, $D_n (n \geq 2)$, and $E_n (n = 6, 7, 8)$, they give dense sphere packing if $n \leq 8$. They can be used as a basis for the efficient block quantizers for uniformly distributed inputs. Since a lattice is a repetition of a structure, every lattice point is surrounded by its Voronoi

region. This region is defined as a space consisting of the nearest neighbors to any given point. Voronoi regions are also called nearest neighbor regions, Wigner-Seitz cells, or Dirichlet regions. If a lattice x is used as a quantizer all the points in the Voronoi region around the lattice point are represented by x (Conway and Sloane, 1982).

3.5.1 Lattice QIM

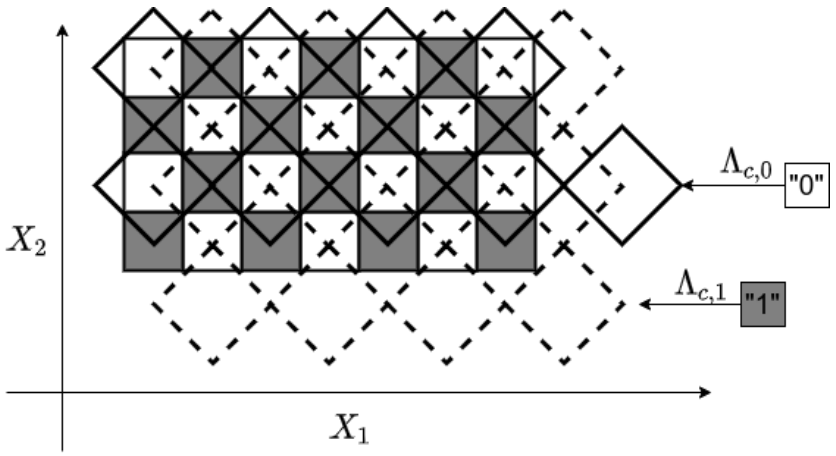


Figure 3.6: Quincunx lattice

We can extend the scalar QIM to L dimensional vector quantizers to form lattice quantizers. Here, for a given lattice Λ_c the quantization function Q can be viewed as a mapping between each point $x \in R^N$ to the nearest lattice point in Λ_c. The Voronoi cell v of the lattice Λ_c, can be defined as $\{x \quad in \ R;\ Q(x) = 0\}$. A more formal definition of a lattice would be an integral combination of the basis vectors $\Lambda = \{x = vG, v \in Z^N\}$, where, G is the generator matrix or the $N \times N$ stack of the lattice basis vectors. The coarse lattice Λ_c can be considered as a sub-lattice or the shifts of partition of Λ. A fine lattice Λ_f is defined as the N-dimensional lattice partition Λ_f/Λ_c, of a nested lattice, where $\Lambda_f, \Lambda_c \subset \Lambda$. If we define a coarse lattice $\Lambda_c = \Delta Z^N$ and a fine lattice as $\Lambda_f = \Delta Z^N \cup (\Delta/2, \cdots, \Delta/2) + \Delta Z^N$ then the Voronoi

cell v becomes an N-dimensional cube $[-\Delta/2, \Delta/2]^N$ whose normalized second order moment is equal to $\Delta^2/12$. Lattice quantizers are superior to scalar quantizers though scalar quantizers are more prevalent due to their simplicity. To explain the way lattice quantization works we can take a look at the two-dimensional Quincunx quantization. If we consider a host signal component $X \in R^2$ such as the target position information generated by an automotive RADAR sensor. The target location is represented by a pair of position co-ordinates in the two-dimensional Euclidean plane $x_1, x_2 \in X$. In general, we can model this data under the assumption that the variables (x_1, x_2) are drawn from an i.i.d Gaussian distribution. In Quincunx quantization, one bit is embedded by modifying the position of a pair of samples (x_1, x_2) as shown in Figure 3.6. In this method, for embedding either '0' or '1' two separate coarse lattices $\Lambda_{c,0}$ and $\Lambda_{c,1}$ are used and they are then obtained by using two different quantizers Q_0 & Q_1. We consider a square Voronoi cell to represent each element of the lattice. The size of Voronoi cell is regulated by the quantization step-size Δ (Zolotavkin and Juhola, 2015).

If we extend this concept to a hexagonal lattice as shown in Figure 3.7, the quantizers associated with the lattice structure can be used to embed two bits into the samples taken from R^2. The quantizers Q_1, Q_2, Q_3 are obtained by perturbing the quantizer Q_0. This can also be viewed as the center of the solid hexagonal lattice $(0, 0)$ shifted over to the coset headers represented by $(01, 10, 11)$ respectively, which can be considered as a set of reconstruction points. It can be observed that in hexagonal lattice QIM or any QIM structure in general, the robustness depends on the distance between the coarse lattice or the center of the solid hexagon in Figure 3.7 and the coset points. Robustness requires the distance to be higher but this will also increase the perceptibility of the watermark. The sphere packing capacity of the lattices provides some ways to find a trade-off between these two parameters. It is proved that the QIM system has high robustness at the cost of a low perceptual

impact together with flexible payload possibilities (Bohó et al., 2013). Now let us

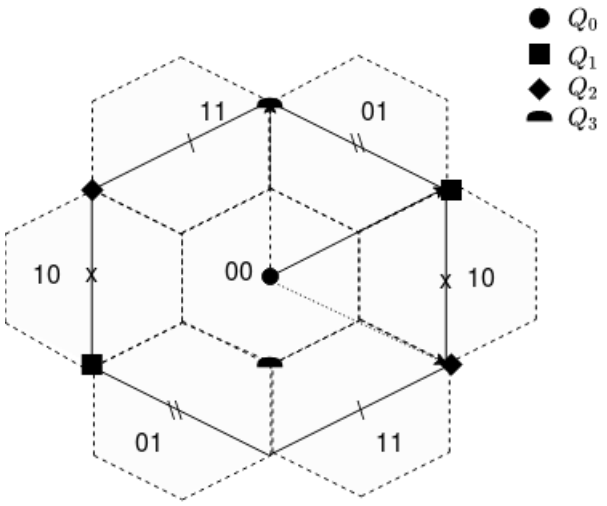


Figure 3.7: Hexagonal lattice QIM scheme

take a look at the QIM properties of lattice quantization. To understand this we take a look at the vector QIM as explained by Chen & Wornell. This scalar QIM concept from Equation (3.11) can be viewed from a lattice point of view where the reproduction levels of quantizers Q_0 and Q_1 show in Figure 3.4 form two cosets of lattices $\Lambda_i,\ i \in [0, 1]$.

$$\Lambda_0 = \frac{-\Delta}{2} + \Delta Z,\ \Lambda_1 = \frac{\Delta}{2} + \Delta Z \tag{3.24}$$

If we replace the scalar quantizer in Equation (3.11) with an L-dimensional Vector Quantizer (VQ) we get L-dimensional lattice quantizer equation. Let's take a look at the simple case of L=2, the VQ is obtained by independently quantizing each co-ordinate shown in Figure 3.6. The lattice into which each co-ordinate gets quantized for a dither size of $\Delta/2$ can be defined as

$$\Lambda_0 = (-\frac{\Delta}{2}, \cdots, -\frac{\Delta}{2}) + \Delta Z^L, \Lambda_1 = (\frac{\Delta}{2}, \cdots, \frac{\Delta}{2}) + \Delta Z^L \tag{3.25}$$

The embedding induced distortion for this VQ is given by $D_1 = \Delta^2/12$ and the rate

of coding is computed as $R = 1/L$. The rate R ranges from $1/L$ to 1 based on the number of bits embedded per dimension. The minimum distance between the cosets of lattices Λ_0 and Λ_1 is given by

$$d_{min} = \frac{1}{2}\Delta\sqrt{L} = \sqrt{3LD_1} \tag{3.26}$$

and the output of the decoder is given by,

$$\hat{m} = \underset{m\in\{0,1\}}{argmin}(\underset{p\in\Lambda_m}{min} |\alpha y - p|) \tag{3.27}$$

If the regular quantization function $Q : R^N \rightarrow \Lambda_c$ is replaced with a dithered quantization function for any given $x \in R^N$ and dither $d \in \nu$ the dither quantization output can be represented as

$$\hat{x} = Q(x - d) + d \qquad \Lambda_c + d \tag{3.28}$$

if the external dither is independent of x, and uniformly distributed over ν then the quantization error $\hat{x} - x$ is also uniformly distributed over ν. If the dither d is shared with the decoder it can be used to randomize the lattice QIM code and provide some level of protection against attacks on the code. Thus, for a given message m and host signal s the dithered lattice QIM equations can be written as

$$u(m) = Q(\alpha s - d_m - d) + d_m + d \qquad \in \Lambda_m \tag{3.29}$$

and the marked sequence can be shown as

$$x = (1 - \alpha)s + u(m) = (1 - \alpha)s + Q(\alpha s) \tag{3.30}$$

the decoders output is given by

$$\hat{m} = \underset{m \in M}{argmin}\ \mathbf{dist}(\alpha \mathbf{y} - \mathbf{d}, \Lambda_m) \tag{3.31}$$

The lattice quantizer results in an M-ary code book, $C = \{(0, \cdots, 0), (\Delta/2, \cdots, \Delta/2)\}$. If length of M grows exponentially with the length of the signal N, the lattice partition Λ_f/Λ_c must satisfy the following properties for better watermarking performance

1. Q must have a bounded mean square distortion D_1 and ν should be a nearly spherical

2. C should be largely spaced

To satisfy the properties listed above, we need Λ_f and Λ_c, to be high dimensional. For all the practical purposes arbitrary high dimensional lattices cannot be built with low computational complexity. To simplify the code generation in higher dimensions, often, the lattices with special structures such as a product of a low-dimensional lattice (cube or cuboid), trellis-coded scalar quantization, or the classical error correction codes like Hamming code are used to define the fine lattice Λ_f.

In the following chapters, we explore the practical applications of the basic 2D and 3D lattice quantizers and how they can be applied to solve the sensor integrity verification problem in the automotive domain.

CHAPTER IV

RADAR Data Integrity Verification-2D QIM

A modern-day vehicle evolved into a cyber-physical system with internal networks (Controller Area Network (CAN), Ethernet, etc.) connecting hundreds of micro-controllers. Starting from the traditional core vehicle functions such as vehicle controls, infotainment, power-train management to the latest developments such as Advanced Driver Assistance Systems (ADAS) and automated driving features, each one of them uses CAN as their communication network backbone. Automated driving and ADAS features rely on data transferred over the CAN network from multiple sensors mounted on the vehicle. Verifying the integrity of the sensor data is essential for the safety and security of occupants and the proper functionality of these applications. Though the CAN interface ensures reliable data transfer, it lacks basic security features, including message authentication, which makes it vulnerable to a wide array of attacks, including spoofing, replay, DoS, etc. Using traditional cryptography-based methods to verify the integrity of data transmitted over CAN interfaces is expected to increase the computational complexity, latency, and overall cost of the system. In this chapter, we propose a light-weight alternative to verify the sensor data integrity for vehicle applications that use the CAN network for data transfers. To this end, a framework for 2-Dimensional Quantization Index Modulation (2D QIM)-based data hiding is proposed to achieve this goal. Using a typical RADAR sensor data transmis-

sion scenario in an autonomous vehicle application, we analyze the performance of the proposed framework to detect and localize the sensor data tampering. The effects of embedding induced distortion on the applications using the RADAR data are studied through a sensor fusion algorithm. It is observed that the proposed framework offers the much-needed data integrity verification without compromising on the quality of sensor fusion data and is implemented with low overall design complexity. This proposed framework can also be used on any physical network interface other than CAN, and it offers traceability to in-vehicle data beyond the scope of the in-vehicle applications.

4.1 Introduction

Since most of the smart sensors as explained in chapter 1, section 1.2.2 use CAN or CAN-FD interfaces for data communication, a vulnerability assessment to cyber-attacks is prudent for these network types. The CAN interface is widely used in the automotive industry due to its robust and fault-tolerant design, however, CAN lacks inherent security to protect against different network attacks. The shortcomings in CAN protocol such as the broadcast message format, clear data transfers, and lack of mechanisms to establish data authentication and confidentiality, expose the CAN network to masquerade attacks, replay attacks, and additional exploits (Lin and Sangiovanni-Vincentelli, 2012). The amount of viable attack vectors on the CAN network has been demonstrated on numerous occasions within automotive security research (Miller and Valasek, 2015). To mitigate known vulnerabilities in CAN, several methods like payload encryption, frame ID-based filtering (Woo et al., 2015), and message authentication code (MAC) calculated based on the payload data are developed. A transport layer security architecture can be built over CAN that can combine both cryptography and MAC to ensure data integrity and authenticity. As more vehicle manufacturers adopt Automotive Open System Architecture (AU-

TOSAR) based platforms for vehicle development, the secure onboard communication (SecOC) is gaining traction, which is again another MAC-based authentication protocol for individual protocol data units (PDU) (AUTOSAR CP Release 4.3.1). When it comes to adopting this traditional cryptography and MAC-based security mechanisms in autonomous vehicles, many practical issues like key management, freshness value handling, and the recovery strategy or how to deal with the failed authentications need to be taken into consideration. Also, the bandwidth and payload length restrictions of the CAN network make it impossible to implement these methods. The legacy systems need to upgrade to CAN-FD that shares the same architecture as CAN but provides more bandwidth due to flexible data rates (Woo et al., 2016). Resolving such practical issues increase the development and maintenance cost of the product as-well-as the overall complexity. To deal with such implementation level

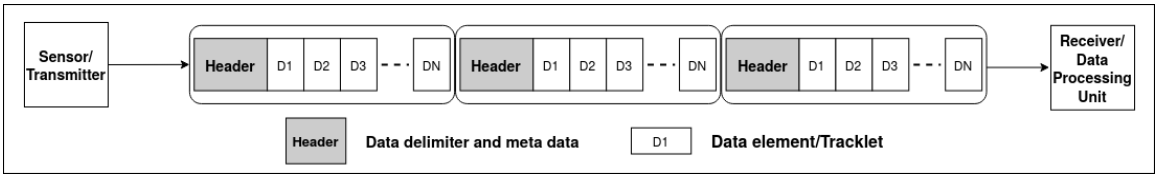


Figure 4.1: Radar data stream

and practical shortcomings of the traditional data integrity verification methods, we introduce a new data hiding based watermarking approach. This approach solves the problem of the data integrity verification in resource-constrained and real-time applications with simple algorithms that do not tax the system with high computational complexity at the same time do not increase the bandwidth requirements of the interface as no additional data is added to the payload. In this method, the watermark is embedded into the sensor using a light-weight software algorithm, which is easy to implement. The concept of using data hiding techniques for sensor data verification was introduced in (Changalvala and Malik, 2019b) for raw sensor data used

in a centralized autonomous driving architecture (Jo et al., 2014). In this chapter, we analyze a different modality; we consider a smart RADAR sensor that produces the processed data. The proposed watermarking method is implemented on the processed RADAR data and its effects on the outcome of a sensor fusion algorithm are verified. With the constraints in the automotive networks as mentioned in the section 1.3 in mind, here we propose a watermarking solution that can work well under these constraints and yet help verify the integrity of the sensor data. The traditional watermarking methods are vulnerable to watermark estimation attacks. To address it, we propose to introduce freshness in the watermark generation process based on the data available on the in-vehicle network such as the GPS timestamp to generate a watermark to be embedded into the sensor data that need to be secured. Sharing the watermark generation algorithm between the sensor and the receiver eliminates the need to exchange the watermarking scheme over any secure channel. Another

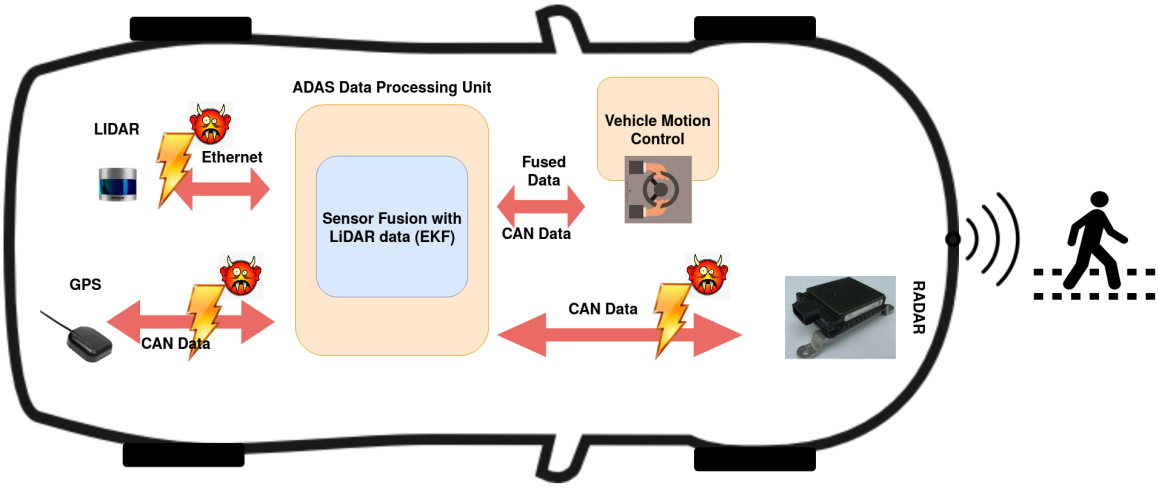


Figure 4.2: Block-diagram of problem statement

aspect to consider while using the data hiding based watermarking techniques is the embedding induced distortion. Through experiments, we provide an analysis on the embedding induced distortion of the 2D QIM method and its impacts on sensor data

and downstream fusion algorithms.

4.2 System & Attack Model

In the system model, we assume that a centralized ADAS unit makes autonomous driving decisions using the data fed by satellite sensors over the vehicle network, as shown in Figure 4.2. The ADAS unit fuses incoming data and performs the necessary information extraction from the object detection lists. This processed information is further used to build autonomous vehicle features like perception and localization (Changalvala and Malik, 2019b). We assume that both the sensors or data origin, and the sink or the central data processing ADAS module are clean. The data input into the system via sensors is authentic. The attacks are launched during the data transmission from the sensor to the ADAS module over the vehicle network, as shown in Figure 4.2. This threat model is more attractive to attackers as the impact factor is high. The damage that can be done by continuously faking or deleting the sensor information as an insider attack is high for autonomous vehicle applications. The system requires a GPS receiver on the vehicle network transmitting timestamp data periodically, and the sensors, along with the ADAS unit have access to this GPS data. The data structure is shown in Figure 4.1 is assumed for the RADAR sensor data. Each data-set starts with a header delimiter that contains information such as the number of tracked objects, unique data identifiers, etc. The header is followed by the stream of data elements themselves that can contain multiple fields based on the capability of each RADAR sensor unit, but for simplicity, we assume the minimum content such as the tracked object Cartesian coordinates (x, y). This position information is used to embed the watermark. The proposed QIM based watermark embedding method works directly on the data; hence it is important to verify that for a given application, the embedding induced distortion does not affect the output of the application consuming the data. In this chapter, we use a sensor fusion algorithm

called Extended Kalman Filter (EKF) that is widely used in autonomous vehicle applications to analyze the embedding induced distortion. The sensor fusion algorithm takes in the RADAR and LiDAR sensor data and outputs the predicted position of the tracked object.

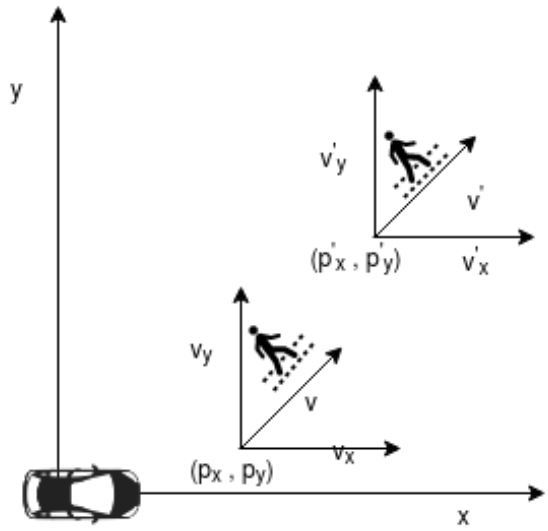


Figure 4.3: State vector for pedestrian motion

4.2.1 Sensor Fusion Data Model

The data-set used to test the proposed framework is generated using the reference code provided by Mercedes Benz autonomous driving utility (Mercedes Benz T., 2018). The data generation scenario is to predict the path taken by a pedestrian walking in front of the vehicle using the on-board sensors, LiDAR, and RADAR, as shown in Figure 4.3. The data-set contains sensor measurements of the location and velocity of the pedestrian. The RADAR sensor measurements are represented in polar coordinates as ρ, ϕ, $\dot{\rho}$, where ρ is the radial measured distance to the target, ϕ is the measured lateral angle to the target, and $\dot{\rho}$ is the rate of change of ρ that results in radial velocity. The Radar measurements are converted from polar to Cartesian

coordinates using the following equations.

$$\begin{aligned} x &= \rho * cos(\phi) \\ y &= \rho * sin(\phi) \end{aligned} \tag{4.1}$$

The LiDAR measurements are represented as position coordinates (x, y). Along with these measurements, the data-set also contains the GPS timestamp at which the data is collected. The time delta between sensor measurements is set to $\delta t = 50$ ms. At each time sample δt, the ground truth values for the pedestrian position and velocity for each sensor (p_x, p_y, v_x, v_y) are calculated based on a constant velocity motion model. The motion model considered for data generation is a 2D bicycle model, and yaw-rate is assumed to be zero, represented by following equations:

$$\begin{aligned} \dot{\theta} &= 0 \\ x^{'} &= x + v \cdot \delta t \cdot cos(\theta) \\ y^{'} &= y + v \cdot \delta t \cdot cos(\theta) \end{aligned} \tag{4.2}$$

where, v is the target velocity, δt is the elapsed time, and θ is the yaw, (x, y) & $(x^{'}, y^{'})$ are the initial and final position values respectively. From these ground truth values,

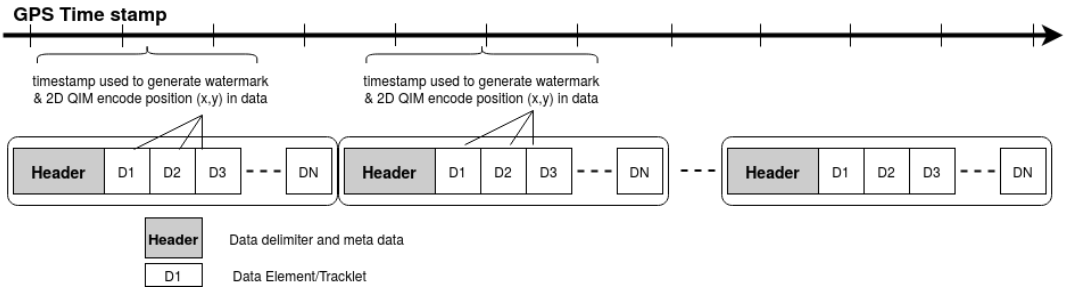


Figure 4.4: Proposed framework and 2D QIM embedding process

the measurement values at each time step are obtained by adding uncertainty in the

form of Gaussian noise of configurable variance. The measured values are represented as the following

$$m_{t_k} = g_{t_k} + \epsilon \tag{4.3}$$

where m_{t_k}, g_{t_k}, represent the measurement and ground truth values respectively at time t_k and ϵ is the measurement error represented by an independent and identical distribution (i.i.d) Gaussian noise with zero mean and covariance matrix $R > 0$, i.e. $\epsilon = N(0, R)$.

4.3 Proposed Framework

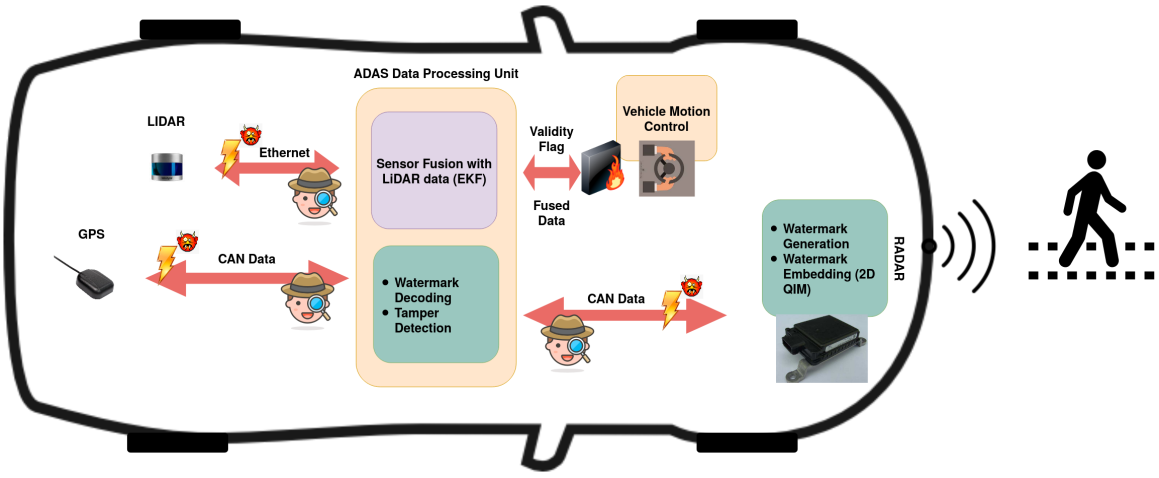


Figure 4.5: Block-diagram of proposed method

In our proposed sensor data integrity verification framework, we assume the following. First, the sensor and the ADAS processing unit share the watermark generation algorithm. Second, the ADAS processing unit collects detection lists from different sensors, in this case, LiDAR and RADAR, and generates a fused list of detection. The proposed framework, as shown in Figure 4.5 is divided into three parts.

- Watermark generation
- Watermark embedding

- Watermark decoding

The watermark generation is done in the sensor using the GPS timestamp information. This binary sequence of message $m_e = f(t_{gps})$ is embedded into the position data of the object detection list from the sensor using 2D QIM data hiding method. This watermarked data is transmitted to the ADAS unit over an in-vehicle communication network, such as CAN. The ADAS unit receives the watermarked data along with the timestamp from the GPS sensor. This data is given as an input to the sensor fusion algorithm as-well-as to the integrity verification algorithm. The integrity verification algorithm that runs in parallel to sensor fusion generates the embedded message sequence m_e using the same method as the sensor. Also, the embedded message sequence from the received frames m_d is extracted using the decoding process. The decoded message sequence is compared against the embedded message sequence to detect and localize the modified data across different attack vectors, as explained in section 4.4. data tampering is detected, a validity flag is set to qualify the fused

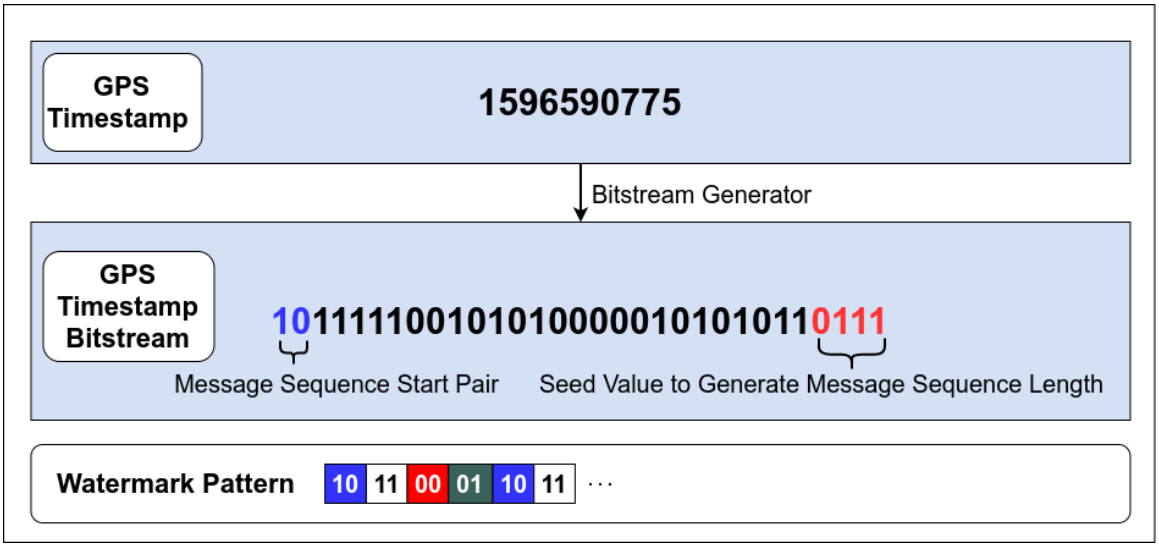


Figure 4.6: Time-stamp conversion

object list. This is represented as a fire-wall between the ADAS unit and the motion control unit in Figure 4.5, which prevents the tampered data from controlling the

vehicle. The data integrity verification mechanism is portrayed as a silent spy in Figure 4.5. It acts in the background unnoticed by the attacker to detect and localize the tampering and alerts the vehicle control algorithm about the integrity of the data.

4.3.1 Watermark Generation

A core component of our integrity verification network is our watermark generation technique. Our proposed model leverages timestamp from GPS sensor data to generate a binary sequence that will be embedded into subsequent RADAR data frames downstream. The diagram in Figure 4.6 represents a visual depiction of a GPS sensor timestamp converted into a bitstream. Assuming an architecture supporting little endianness, the two least significant bits (LSB) and the most significant bit (MSB) nibble are parsed and stored in a secured buffer, to be utilized by the sequence generator described in Algorithm 4. The LSB pair stored is used to determine the starting bit pair for the generated sequence. This adds a level of obfuscation to the generated sequence by changing the starting bit pair of the binary sequence to be embedded in the available data elements. In addition, the MSB nibble is converted to its decimal representation and utilized as a seed value to generate a random number to fall within the theoretical maximum for potential data elements generated in one message payload.

The sequence generator represented by Algorithm 4 will utilize the previously gathered information from the GPS time stamp to generate a deterministic sequence. This process involves taking the range limited random number x, generated from the seed value of the MSB nibble, where: $num_{Pairs} = \lfloor x \rfloor$ to determine the length of the sequence. A two-bit value is then incremented and appended to the generated sequence buffer. The proposed 2D QIM embedding method allows for a message sequence of integer values in range ($0 \leq m_{val} < 4$), hence the generated two-bit value is modulated by 4, to keep it within the allowed range. The desired length

of the sequence is, by design, dependent on the seed value calculated from parsing the GPS timestamp. If the sequence is shorter than the amount of data elements in the message payload, the generated sequence will be reused. The randomness in this generated message sequence comes in the form of the start pattern and the length of the message sequence both of which can be recreated by the receiver using the same GPS timestamp message.

4.3.2 Watermark Embedding

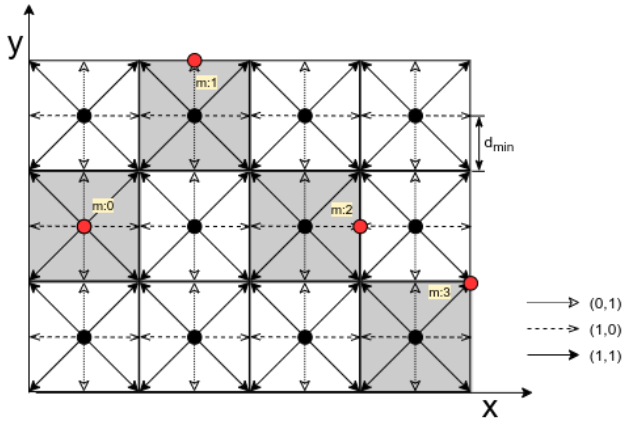


Figure 4.7: 2D QIM scheme

We use a 2D QIM-based data hiding method for watermark embedding. Here we use simple cubic lattice QIM approach mentioned in section 3.5. If we use regular quantization function in Equation (3.28), the resulting watermarked signal s_w for a 2D QIM can be represented as:

$$s_w(s_{c_k}, m_k) = q_{m_k}(s_{c_k}, \Delta) \tag{4.4}$$

where $q_{m_k}(\cdot)$, denotes 2D QIM quantizer which is expressed as:

$$q_{m_k}(c_k, \Delta) = \text{round}\left(\frac{c_k}{\Delta}\right) \cdot \Delta \pm \frac{\Delta}{2} \cdot m_k \tag{4.5}$$

It can be observed from Equation (4.5) that the host signal gets modified after the data embedding and the distortion level is proportional to the perturbation. This feature provides the flexibility to select a distortion level that works for a particular end application. This motivated us to select QIM over other available watermarking methods. In QIM, the quantization operation uses a unique set of quantizers that result in a reconstruction grid (Joachim and Bernd, 2002). The dimensionality of the reconstruction grid depends on the message symbol size. If a message has an n-dimensional symbol, it results in a $\log_2(n)$-dimensional reconstruction grid. For example, a binary message $m \in \{0, 1\}$, where $n = 2$ results in a 1D reconstruction grid. If we extend this concept to a two-dimensional data-set like RADAR data-position vector (x, y), a four-dimensional message symbol $m \in \{0, 1, 2, 3\}$ can be used to hide data resulting in a 2D reconstruction grid. Each corner of this grid can be considered as a reconstruction point in recovering the embedded message. In Figure 4.7 a sample data set is depicted with different embedded message symbols. After the initial quantization, the position data points that fall within the highlighted polygons are represented by the center of the regular polygon or the black dot $c_k = \{x_k, y_k\}$. In 2D QIM, based on the embedded message symbol $m_k = \{m_{xk}, m_{yk}\}$, this center point is moved to one of the eight fixed locations on the boundary of the polygon as represented by the red dot. The minimum value of the separation distance between the reconstruction points d_{min}, determines the resilience of the framework to the channel noise. This is a configurable parameter in QIM based watermarking that comes as an advantage when trying to adapt this framework to different end applications. Another advantage being the host-signal interference rejection because of the non-intersecting reconstruction points (Chen and Wornell, 1998).

In the proposed framework, a simple algorithm parses through the generated message symbols and applies the corresponding quantizer to the RADAR position data as per Equation (4.5). The sender uses the GPS timestamp from a pre-defined in-

terval and is based on the procedure explained in section 4.3.1; the watermark is generated and embedded into the data elements of the RADAR data as shown in Figure 4.4. The modified data is transmitted over the selected data transfer interface like CAN/CAN-FD with additional meta-data included in the header that acts as a delimiter to the data-elements. The watermark stays with the data irrespective of the data-link or transport protocols used to send the data to the receiver. Also, since there is no additional data added in the form of MAC, the interface bandwidth requirements remain the same.

4.3.3 Watermark Decoding

The RADAR detection object list data received by the ADAS unit can be directly used by the sensor fusion algorithm. Tamper detection and localization algorithms can run in parallel. This is a significant advantage of the watermarking method over any other encryption-based methods. The decoding algorithm works similarly to the embedding. Each received position value is quantized using all the different quantizers used for embedding to generate different reconstruction points, which is a set of four in our case. These four reconstruction point values are compared with the received value, and the reconstruction point that returns the least difference value, as shown in Equation (4.6) is considered the decoded message.

$$m_d = \underset{m \in \{0,1\}}{argmin} |s_w^{'}(s_i, m_d) - s_w(s_i, m_i)| \tag{4.6}$$

where, $s_w^{'}(s_i, m_d)$ represents a distorted received signal, m_i is the embedded message and m_d is the decoded message. The decoding step also regenerates the embedded sequence following the same procedure explained in section 4.3.1 as it receives the same GPS timestamp over CAN. Here it is assumed that the sensors are time-synchronized by the universal timestamp provider such as a GPS sensor (AUTOSAR CP R19-11).

4.4 Security Analysis & Performance Evaluation

Successful attacks are the ones that go undetected by the detection framework. In this section, we discuss various attack scenarios possible if an attacker gets access

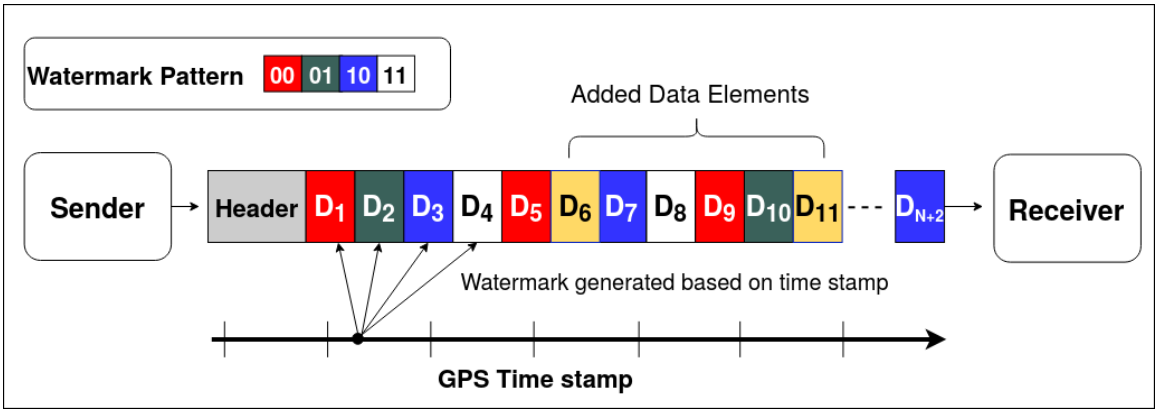


Figure 4.8: Data addition attack vector depiction

to the vehicle network and how the proposed framework can detect and localize these attacks. As a part of the attack model, we assume that the attacker has a good knowledge of the vehicle network protocols and automotive electrical system architecture. He has tools available to sniff the vehicle network and replay the modified messages on CAN/CAN-FD. With this knowledge, we identified three ways the attacker can modify the sensor data once he sniffs it from within the network. The attacks are broadly classified as

- Data addition
- Data deletion
- Data modification

for each of these attack scenarios in the following sections, we analyze how the proposed framework performs.

4.4.1 Data Addition

Data addition is an attack scenario where the attacker modifies the RADAR reflections or tracklets with additional fake data elements either by copying the existing elements or by adding completely random data. A typical add attack scenario is represented in Figure 4.8, the data elements D_6 and D_{11} are added to the original sequence of the RADAR data, increasing the total count of elements from n to $n+2$. With the proposed framework, position information (x, y) is 2D QIM encoded by the sender with a message pattern generated based on the GPS timestamp, as explained in section 4.3.1. This message pattern is represented by color-code (Red, Green, Blue, and White) in Figure 4.8. Here, for simplicity, we assume a fixed pattern length of four but the framework can accommodate variable length patterns. With the additional data elements added during an attack, even if they are a copy of the existing data elements, the encoded sequence gets disrupted. The receiver expects a message sequence of Green for data element D_6 and Blue for data element D_{11}, but the algorithm detects the subsequently received elements do not have the expected message sequence. Here, we assume that the receiver knows the length of expected data elements. The sender and receiver can agree on a pre-defined range of values for the length or have an increment counter in each data-element, etc. to get the length. Knowing the encoded data element length along with the side information received from a different sensor modality like the GPS timestamp, as mentioned in section 4.3.1, will help the decoder to generate the encoded message pattern. Based on the lengths of the encoded message sequence l_{encode} and the decoded message sequence l_{decode}, the type of attack can be determined, i,e it can be determined that the elements are added if $l_{decode} > l_{encode}$. As shown in Algorithm 3, the decoded message sequence is compared with the expected message sequence in an $O(N)$ loop to find out the location of newly added data elements. This algorithm assumes that the added element's pattern is different from its adjacent element. Figure 4.11 de-

picts the performance of the algorithm in the presence of additive uniform noise. The results show the robustness of the tamper detection algorithm and the proposed data-hiding based framework performance in the presence of channel noise. The QIM based methods can recover the watermark as long as the channel noise is confined to the below equation.

$$d_{min}^2 > 4 \cdot N \cdot \sigma_n^2 \tag{4.7}$$

where σ is the standard deviation of channel noise, and N represents the number of encoding bits or dimensions, and d_{min} represents the minimum distance between the reconstruction points (Chen and Wornell, 1998). It can be observed from Figure 4.11 that the proposed framework can detect and localize the tampered data elements with 100% accuracy when the noise is within bounds as per Equation (4.7), for a given step-size of $\Delta = 1\ cm$.

4.4.2 Data Deletion

In this scenario, as shown in Figure 4.9, the attacker modifies the RADAR detections either by carefully eliminating chosen targets or by deleting random elements. A typical delete attack scenario is represented in Figure 4.9, the data elements D_6 and D_{11} are deleted from the original sequence of the RADAR data. This decreases the total count of elements from n to $n-2$. In the proposed method, the sender embeds the message pattern generated from the GPS timestamp in the position information (x, y) of the data elements. The message pattern is represented by color-code (Red, Green, Blue, and White) in Figure 4.9. When the data elements get deleted, the message embedding sequence gets disrupted. The receiver expects a message sequence of Green for data element D_6 and Blue for data element D_{11}, but it can detect that the received elements D_7 and D_{12} respectively do not have the expected pattern. Based on the lengths of the encoded message sequence l_{encode} and the decoded message sequence l_{decode}, the type of attack can be determined, i,e it can be determined that the

elements are deleted if $l_{decode} < l_{encode}$.

The tamper localization algorithm is shown in Algorithm 2. The decoded message sequence is compared with the expected message sequence in an $O(N)$ loop to determine the location of the deleted data elements. The results of the algorithm are shown in Figure 4.11. The algorithm detects and localizes the delete attack vector with 100% accuracy as long as the noise is bounded by Equation (4.7). It can be observed from Figure 4.11 that as the noise variance increases, the accuracy falls for a given step-size.

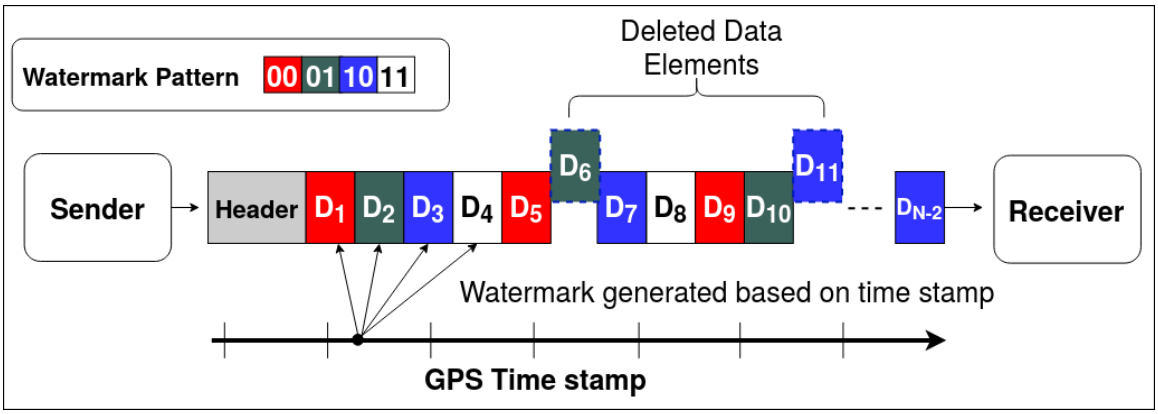


Figure 4.9: Data deletion attack vector depiction

4.4.3 Data Modification

During a data modification attack, as shown in Figure 4.10, the attacker modifies the RADAR detections by altering the existing data element content. Figure 4.10 represents a typical data modification attack. The data elements D_6 and D_{11} are modified in the original sequence of the RADAR data. This type of attack does not change the total count of elements. In the proposed method, the sender embeds a message pattern generated from GPS timestamp, as explained in section 4.3.1, into the position information (x, y) of the data elements. The message pattern is represented by color-code (Red, Green, Blue, and White) in Figure 4.10. When the data elements

get modified, the embedded message sequence gets disrupted. The receiver expects a message sequence of Green for data element D_6 and Blue for data element D_{11} and detects that the received data elements do not have the expected pattern. To get the location of the modified data elements, as shown in Algorithm 1, the decoded message sequence is compared with the expected message sequence in an $O(N)$ loop. The accuracy of the algorithm is represented in Figure 4.11. This algorithm assumes that random elements are modified within a given message pattern, and the channel noise is less than the step size. It is observed from Figure 4.11 that as the noise variance increases, the localization accuracy of the algorithm decreases. However, when the noise is within bounds, the detection and localization accuracy of the modified data elements is 100%. Similar to the trend observed in the other two algorithms.

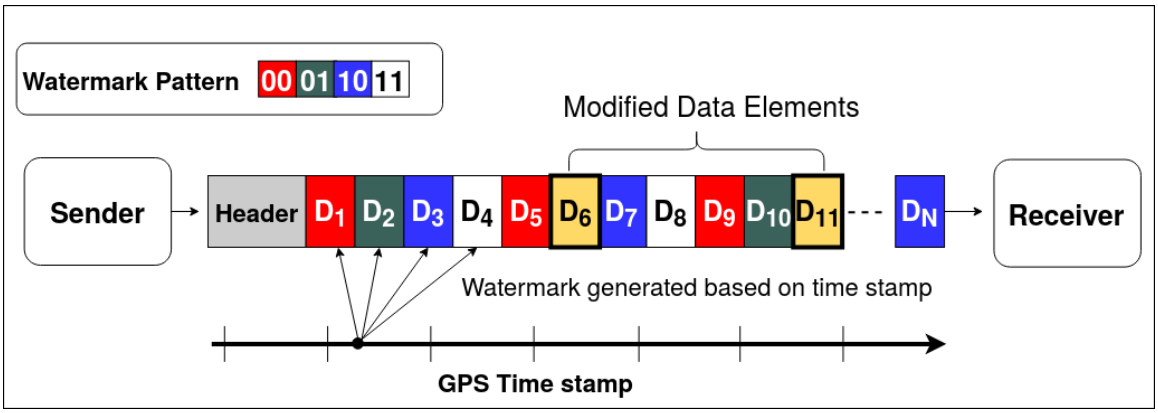


Figure 4.10: Data modification attack vector depiction

4.5 Experiments & Results

4.5.1 Impact of Embedding Distortion on Object Detection

Though the data-hiding based sensor integrity framework is computationally less complex than the traditional cryptography, one of the major concerns with the method is the embedding induced distortion as the watermark is embedded directly

into the data by altering it. A question arises whether the end application or the consumer of this data can handle this distortion. An answer to this question would help the automotive industry adopt the watermarking techniques for sensor data or other data integrity verification applications. In this study, we analyze the effects of embedding RADAR data using the proposed 2D QIM data hiding framework on a sensor fusion algorithm. The sensor fusion method we use for this study is EKF. In autonomous vehicles, Kalman filters are used to estimate the state of any dynamic system, such as position estimation of moving objects on the road. In doing so, the Kalman filter only needs current observations and previous predictions, hence Kalman filter is a light-weight fusion algorithm (Jetto et al., 1999), (Rigatos, 2010), (Madhavan and Schlenoff, 2003). They are also good at handling the measurement inaccuracies in the sensors, i.e., sensor noise. The EKF based sensor fusion algorithm takes inputs from two or more sensors and generates a combined prediction of the tracked object at every time step. The impact of the proposed 2D QIM method for sensor data integrity verification is estimated based on the output of the EKF. As explained in section 4.2.1, in this experiment, we use measurements from two onboard vehicle sensors LiDAR and RADAR to estimate the state of a pedestrian moving in-front of the car. The same object, in this case, a pedestrian, will be detected by the two sensors, and a Kalman filter fuses the data and predicts the accurate position of the pedestrian.

To predict the position of a target, a Kalman filter uses a motion or process model that estimates the future location of the target or object of interest. In this chapter, we use a constant velocity motion model as a baseline for target motion estimation. The motion model, as depicted in Figure 4.3, predicts the position and velocity of the pedestrian at a future time t_{k+1}, based on the values at time-step t_k. This position information is provided as a 2D position and velocity vector called the state vector. In the context of this chapter, the state vector consists of the predicted position and

velocity of the pedestrian represented as

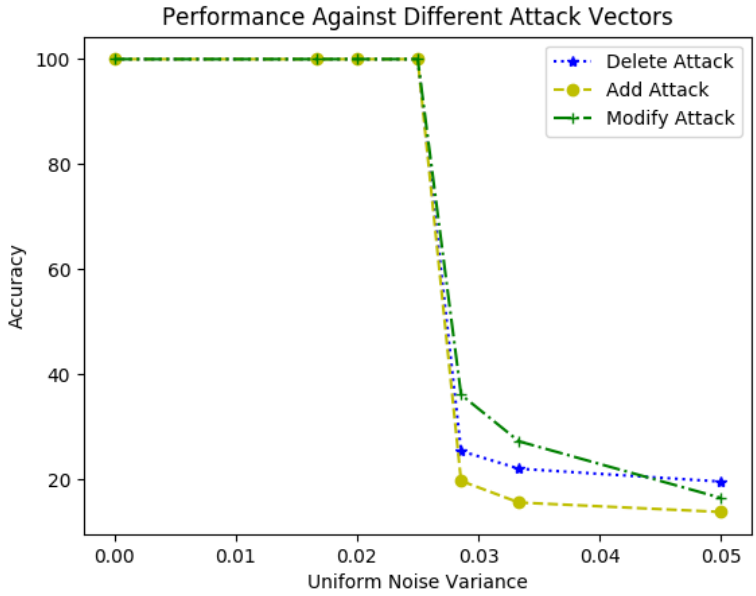


Figure 4.11: Tamper localization algorithm performance under varying channel noise

$$x = [p_x \quad p_y \quad v_x \quad v_y]^T \tag{4.8}$$

where (p_x, p_y) are (x, y) components of pedestrian position and (v_x, v_y) are (x, y) components of his velocity at a given time step t_k. The Kalman filter consists of prediction and update steps. In the prediction step, the state vector $x^{'}$ at next time step t_k is estimated along with the uncertainty $P^{'}$ based on values of x and P at previous time step t_{k-1} and the motion model. During the update step, for every new measurement at time t_k, the estimation function performs the measurement update. The deterministic part of the prediction step F is the state transition matrix. The uncertainty measure P is a stochastic process modeled as random noise that affects the prediction step. The state vector $x^{'}$ can be estimated as

$$x^{'} = f(x, \mu) \tag{4.9}$$

where, μ is the stochastic part, represented as $N(0, Q)$, this can be re-written as

$$x^{'} = Fx + \mu$$
$$P^{'} = FPF^{T} + Q \tag{4.10}$$

where F is the state transition matrix that models state transitions from previous

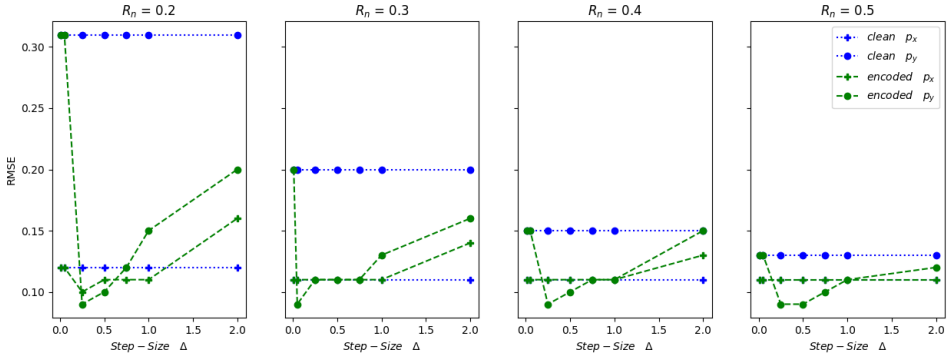


Figure 4.12: RMSE comparison at $R_m = 0.4$

time step t_{k-1} to current time t_k, μ is the added noise, Q is the process co-variance matrix that models the stochastic part of the state transition function. A linear motion model with constant velocity is used to define the state transition matrix F. The position at next time step t_k is given by

$$p^{'}_{t_k} = p_{t_{k-1}} + v_{t_{k-1}} * \delta t \tag{4.11}$$

where $\delta t = t_k - t_{k-1}$ and since the model assumes constant velocity, the velocity at next time step is given as

$$v^{'}_{t_k} = v_{t_{k-1}} \tag{4.12}$$

based on the above model, the Kalman filter uses the estimated state to predict the

pedestrian position. In the update step, the sensor measurements are used to correct

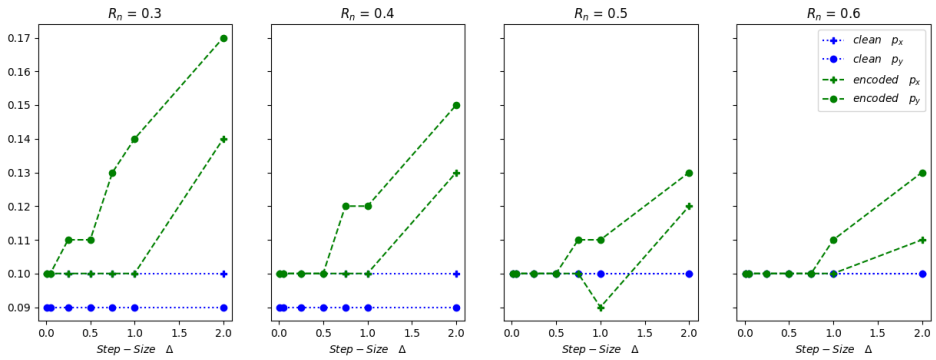


Figure 4.13: RMSE comparison at $R_m = 0.5$

the predicted states and to obtain more accurate estimates. In the measurement function, the vehicle only senses the pedestrian position and can be expressed as

$$z = [p^{'}]^T \tag{4.13}$$

The measurement step that precedes the update step relies on the measurement model, measurement matrix H, and covariance matrix R to correctly estimate the measurement vector z. The measurement matrix is required to transform the measurement vector z to the state vector as shown in Equation (4.8). The measurement function can be represented as

$$z = Hx + \omega \tag{4.14}$$

where H is the measurement matrix that projects object position belief into the measurement space of the sensor and ω is the measurement error that encompasses all the uncertainties in measurements from the sensor represented as a Gaussian with zero mean and covariance matrix R, $\omega \approx N(0, R)$. Assuming the measurement components are not cross-correlated, the covariance matrix R becomes a diagonal

matrix. The dimensionality of R depends on the size of the measurement vector z, which is two for LiDAR and three for RADAR in our case. Hence, R becomes a $3X3$ diagonal matrix for RADAR and a $2X2$ diagonal matrix for LiDAR. The measurement matrix H also differs based on the sensors used by the fusion algorithm. Since LiDAR measures the position of the target in the Cartesian coordinates (x, y), the state vector to measurement vector transition is linear, and calculation of the measurement matrix H is straightforward. It just needs to discard the velocity from the state vector. Hence during the update step, standard Kalman filter transitions are applied for LiDAR measurements. In the case of RADAR, the transition is non-linear as RADAR measures ρ, ϕ, $\dot{\rho}$ of the target. During the update step, to handle the non-linear measurement functions for RADAR measurements, we use the EKF concept. Kalman filters are linear estimators, the extension of this idea to non-linear systems is called extended Kalman filter (EKF) (Madhavan and Schlenoff, 2003). In EKF, the non-linear state and observation equations are linearized using Jacobian matrices. Hence for RADAR sensor Jacobian of H is computed to get the linear approximation. Once the z is computed, the update step or correction step is performed where the latest measurements are used to update the state estimates and their uncertainties as following:

$$y = z - Hx^{'} \tag{4.15}$$

Here y is the error value or the difference between the prediction and actual measurement at a given time step. The estimation error S is computed as

$$S = HP^{'}H^{T} + R \tag{4.16}$$

The Kalman gain K is computed as

$$K = P^{'} H^{T} S^{-1} \tag{4.17}$$

After the computation of the Kalman gain, the predictions are updated using the following equations and these steps are repeated for the entire drive cycle.

$$x = x^{'} + Ky \tag{4.18}$$

$$P = (I - KH)P^{'} \tag{4.19}$$

It can be observed that in an EKF, the uncertainty in both the process and measurements are taken into consideration. In general, the measurement uncertainty or the measurement noise covariance matrix R in Equation (4.14), is the inherent sensor behavior and hence provided by the sensor manufacturer. Whereas, the process uncertainty Q in Equation (4.10), is defined based on the motion model and other application related assumptions. If we use the EKF as the sensor fusion algorithm, three configuration parameters can affect the algorithm outcome in the proposed framework. The first being the measurement noise matrix R, the second the overall process noise Q, and the third, the embedding step-size Δ. In this experiment, we analyze the impact of the watermark embedding using the 2D QIM method on the sensor fusion algorithm's output under different configuration scenarios. We use two types of RADAR data as an input to the EKF algorithm that predicts the state vector of the pedestrian. The first type is the clean and unmodified data, and the second type is 2D QIM modified data. The resulting predictions from EKF are compared against the ground truth position vectors in both the cases using the Root Mean

Square Error (RMSE) metric calculated as following

$$RMSE = \sqrt{\frac{1}{n}\sum_{t=1}^{n}(x_t^{gt} - x_t^{pred})^2} \tag{4.20}$$

where x_t^{gt} & x_t^{pred} are the ground truth and predicted position vectors respectively at a given time t and n is the length of data. The RMSE value is used to determine the accuracy of the prediction. Low RMSE value indicates that the sensor fusion algorithm predicted the tracked object position accurately throughout the track path. The RMSE values of position vector (p_x, p_y) predictions generated from clean and watermarked inputs to the EKF are shown in Figure 4.12 & 4.13.

The measurement input to EKF has a measurement noise component which is dependent on the intrinsic electronic characteristics of the sensor. This can be represented as an additive Gaussian noise ω as shown in Equation (4.14). The measurement noise covariance R represents the deviation of the sensor measured values from the true values. This deviation is estimated during the calibration phase by the sensor manufacturer. In the absence of the sensor manufacturer data, it can also be estimated using analytical methods (Park et al., 2019). To accurately compensate for the measurement noise in an EKF the R value for a given sensor must be known or estimated to be used in Equation (4.14).

If we consider $\omega_{R_m} \approx N(0, R_m)$ as the known or measured measurement uncertainty and $\omega_{R_n} \approx N(0, R_n)$ as the overall measurement uncertainty used in the sensor fusion EKF algorithm in Equation (4.14), an EKF provides accurate predictions when the value of $R_n \geq R_m$. Here, it is always better to keep the R_n & R_m values close to each other. If the EKF requires an inflated R_n value to incur correct predictions then it could be concealing other issues in the measurements like measurement outliers and non-Gaussian nature of the noise. The measurement uncertainty values used in the EKF ω_{R_n} can be represented as a combination of two

or more different noise distributions with data satisfying the i.i.d criteria. Let's say, $\omega_{R_n^1} \approx N(0, R_n^1)$ and $\omega_{R_n^2} \approx N(0, R_n^2)$ are two different noise distributions that contributed to the overall noise ω_{R_n}, then the resulting distribution can be represented as:

$$\omega_{R_n} = N(0, R_n^1 + R_n^2) \tag{4.21}$$

The RMSE results depicted in Figure 4.12 & 4.13 show that the 2D QIM embedded RADAR data can be considered as an added random noise contributor to the overall measurement uncertainty and it can be represented by R_n^1 or R_n^2 in Equation (4.21). In this experiment, the RMSE values for clean and 2D QIM embedded RADAR data are calculated at different measurement noise covariance matrix values $R_m \in (0.4, 0.5)$, $R_n \in (0.2, 0.3, 0.4, 0.5, 0.6)$ and varying embedded step sizes $\Delta \in (0.01, 0.05, 0.25, 0.50, 0.75, 1, 2)\ m$. Considering the R_m as the measurement error covariance provided by the sensor manufacturer, the EKF which accepts this RADAR sensor data should use a covariance matrix value R_n above or equal to the R_m uncertainty. It can be observed from Figure 4.12, when $R_n \geq R_m$, the RMSE values of position vector for the 2D QIM embedded data is less than or equal to the RMSE values from clean data for step-size $\Delta < 0.75\ m$. With a given range of $p_x \approx 18.5\ m$ & $p_y \approx 12.5\ m$ in the data under test, the results show that the fusion algorithm can recover from position data perturbations of up-to 6%. As the R_n value goes below the R_m, the RMSE of encoded data is less than the clean data only when $\Delta < 0.05\ m$. This shows that the embedding induced distortion at higher step sizes is acting like additional uncompensated noise and introduces prediction errors. Similar results are observed for the state vector predictions in case of data with measurement covariance matrix value $R_m = 0.5$, as shown in Figure 4.13. It can be inferred from the results that in the case where the measurement covariance $R_n < R_m$, as the embedding step-size increases, the measurement noise value increases, and hence the predictions of the embedded data elements are off. But as the

R_n value is increased above the R_m, the embedding induced distortion is gracefully handled by the fusion algorithm, and we observe low RMSE values even at larger step sizes. This phenomenon can be explained by Equation (4.21). Here the embedded induced distortion acts like an additive Gaussian noise component. The inherent randomness in the watermark generation and embedding, which acts as noise, adds up to the randomness in the sensor noise. These two noises are independent of each other; hence the resultant effect is additive. This increases the RMSE value of the prediction error when the fusion algorithm does not consider and compensate for this additional noise. These experiments, when repeated at different permissible values of process noise covariance matrix $Q > 0$, showed similar results.

Apart from the embedding induced distortion analysis, two different experiments are conducted to measure the other performance parameters of the detection framework, such as the bit error rate and the false alarm rate.

4.5.2 Bit Error Rate

In this experiment we analyze the errors in the decoded bit stream in the presence of channel noise. The decoder step in the proposed framework generates a binary message stream $M_{x,y} = \{m^1_{x,y}, m^2_{x,y}, \cdots, m^N_{x,y}\}$, from the RADAR data elements. The bit error rate BER is calculated by comparing each bit in the decoded message $m^i_{x,y} \in \{m^i_x, m^i_y\}$ with the embedded bit $\hat{m}^i_{x,y}$ as follows:

$$BER = \frac{\sum_{i=1}^{n} \mathbf{I}_{m^i_{x,y} \neq \hat{m}^i_{x,y}}}{n} \tag{4.22}$$

where $\mathbf{I}$ is the indicator function, and n is the size of the decoded message bitstream. When no additional noise is added to the RADAR data elements, the BER is close to 8.6%, which corresponds to the noise due to the attack vectors. As the channel noise modeled by an uniform distribution is added to the data, the BER stays below 9.5% for the noise variance $\sigma < \Delta/5.65$, for a given step-size Δ. As the noise vari-

ance increases beyond the threshold in Equation (4.7), the BER value increases as shown in Table. 4.1. It can be observed that the robustness to the channel noise is directly proportional to the step-size Δ, which-in turn is directly proportional to the embedding induced distortion.

Noise variance σ	BER %	FalseAlarm %
0.0	8.6	0.0
$\Delta/6$	9.2	0.0
$\Delta/5$	9.2	0.0
$\Delta/4$	8.6	0.0
$\Delta/3.5$	18.6	61.1
$\Delta/3$	28.6	75.0
$\Delta/2$	56.4	85.2

Table 4.1: BER and False-Alarm rate at different noise levels

4.5.3 False-alarm Rate Analysis

The false alarm rate analysis is another important performance indicator of the proposed framework. It measures the number of data-elements the framework determines as tampered when it is provided with clean or unmodified data. In this experiment, the framework is tested with a combination of clean and modified data elements, and the false alarm rate $f_{AlarmRate}$ is calculated as follows:

$$f_{AlarmRate} = N_{FalsePositive}/N_{DataElements} \tag{4.23}$$

where, $N_{FalsePositive}$ is the number of data elements the framework falsely classified as tampered with and $N_{DataElements}$ is the total number of data elements tested. The experiment is repeated at different levels of the additive uniform noise to replicate the channel noise. The results are shown in Table 4.1. It can be observed that the $f_{AlarmRate}$ stayed at 0% when the uniform noise variance $\sigma < d_{min}/(2 * \sqrt(N))$, where $d_{min} = \Delta/2$, $N = 2$ in our framework. As the noise variance increases beyond this threshold the false positives increase resulting in a higher false alarm rate. It

can be concluded from these results that when the channel noise is within acceptable bounds, our framework can achieve 100% detection accuracy with zero false positives. To summarize the findings, the proposed 2D QIM based integrity verification pipeline

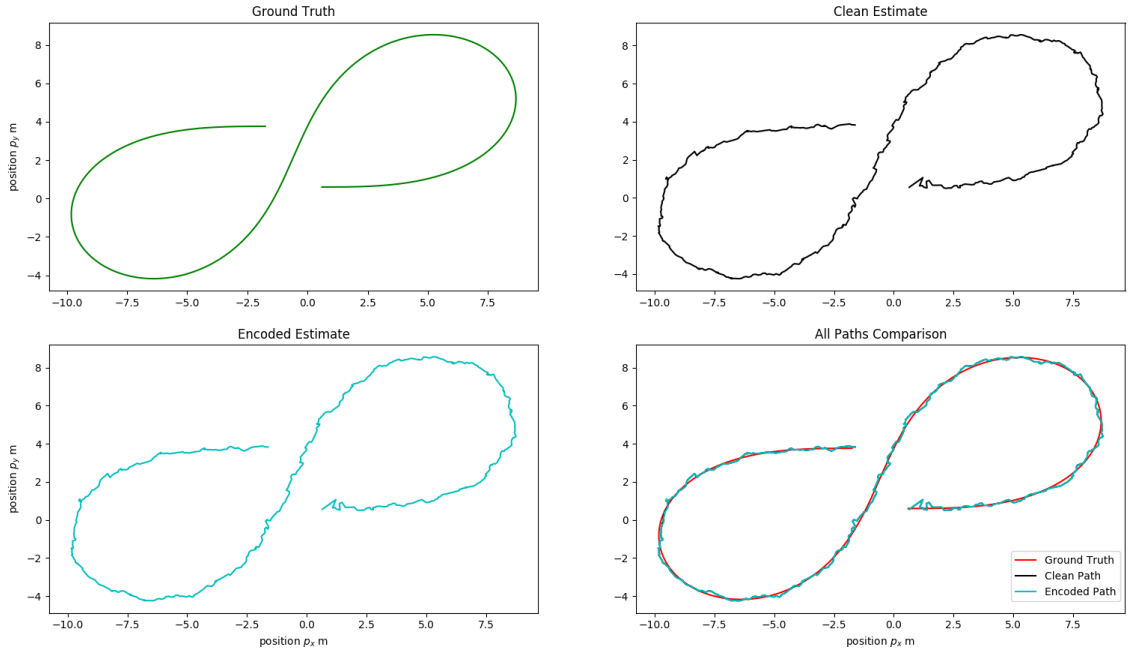


Figure 4.14: Comparison: EKF path prediction from clean and encoded data at R_m = 0.5, R_n = 0.5 & Δ = 0.01 m

is tested for the affects of embedding induced distortion using simulated RADAR data on an EKF based sensor fusion algorithm. The experimental results conclude that the 2D QIM method for watermarking has a little or no effect on the EKF predictions for small values of quantization step-size $\Delta \leq 0.05\ m$, which can be attributed to the minimal distortion induced by the 2D QIM process. A visual representation of the tracked path by sensor fusion EKF algorithm for both plain and encoded inputs at a small step-size $\Delta = 0.01\ m$ is displayed in Figure. 4.14. It can be observed that the predicted state-vectors for both the plain and 2D QIM embedded inputs are similar even when the actual measurement noise covariance R_m and the EKF considered noise covariance R_n are same. As the step size increases, the overall measurement

noise covariance used in the EKF, R_n need to take into account the noise generated by the 2D QIM embedding to get accurate results, this phenomenon is shown in Figure. 4.12, 4.13. Other experiments to measure the tamper localization accuracy and noise resilience of the proposed framework show that the proposed framework works well if the channel noise of the interface is within theoretical bounds presented in Equation. 4.7. The tamper localization accuracy of our framework is close to 100% when the interface noise is zero.

CHAPTER V

LiDAR Data Integrity Verification-3D QIM

Deterministic perception of the surrounding environment is both crucial and a challenging task for autonomous vehicles. A wide range of sensors, including LiDAR, RADAR, cameras, and so on, are used to build the perception layer of an autonomous vehicle. Many interfaces, such as OBD-II, Wi-Fi, Bluetooth, cellular networks, etc., have been introduced in autonomous vehicles to control various functionalities, including V2X communications, over-the-air updates, security, remote vehicle-health monitoring, and so on. These interfaces are introducing new attack surfaces that can be exploited via external as well as internal attacks. Attackers have successfully demonstrated how to exploit these attack surfaces by crafting attack vectors to launch both insider and external attacks. The sensor and sensor data are also vulnerable to both external and insider attacks. Developing safeguards against these attacks is a steppingstone toward the design and development of reliable autonomous vehicles. For instance, failure to detect and localize sensor data tampering can result in an erroneous perception of the environment and lead to wrong path-planning and control decisions. We propose a novel semi-fragile data hiding-based technique for real-time sensor data integrity verification and tamper detection and localization. Specifically, the proposed data hiding-based method relies on 3-dimensional quantization index modulation (QIM)-based data hiding to insert a binary watermark into the LiDAR

data at the sensing layer, which is used for integrity verification and tamper detection and localization at the decision-making unit, e.g., the advanced driver assistance system (ADAS). The performance of the proposed scheme is evaluated on a benchmarking LiDAR dataset. The impact of information hiding on the object-recognition algorithm is also evaluated. Experimental results indicate that the proposed method can successfully detect and localize data tampering attacks, such as fake object insertion (FOI) and target object deletion (TOD). Robustness to noise-addition attacks is also evaluated.

5.1 Introduction

Sensors feed the sensor fusion core in an autonomous vehicle with environment data. If the input to the sensor fusion core is compromised, the resulting decisions down the understand-and-act pipeline would be erroneous and could result in significant damage. Though some redundancy could be built throughout the system by fusing different sensor information, the computational cost of path planning and other control algorithms to work around and ignore the tampered sensor data is much higher than detecting the tampering at the sensor level. It can be observed from Figure 1.2 that given a vehicle architecture in which a sensor transmits raw data to the vehicle for data interpretation, an attacker can exploit vehicle attack surfaces to tamper with the raw sensor data by simple operations like fake object insertion (FOI) or target object deletion (TOD) to dupe the object information extractor and the perception estimation applications that are down the pipeline. By inserting tampered data containing fake objects or by deleting existing objects an attacker can influence the perception and localization algorithms to consider and act on the tampered data. This would result in the ADAS making wrong control decisions like decelerating or braking when it is not supposed to or driving right into a target object. These wrong control decisions can pose a serious safety threat to the occupants of an autonomous

vehicle. *Integrity verification of sensor data before acting on it is crucial.*

To demonstrate the effectiveness of the data hiding-based integrity verification methods for sensor data, a 3-dimensional quantization index modulation (3D-QIM) is implemented on LiDAR sensor point cloud data. Simplicity, low embedding/decoding complexity, quantifiable embedding distortion as a function of embedding parameter Δ, and detection performance as a function of channel distortion and embedding parameters are the salient features of QIM-based data hiding, which is the main motivation behind selecting it over the other information-hiding methods (Malik et al., 2008), (Chen and Wornell, 2001). The proposed method could be easily adapted into other 3D point-cloud data generators like RADAR, red-green-blue-depth (RGBD) cameras, etc.

5.2 LiDAR Point Cloud: Applications

The LiDAR sensor plays a key role in an autonomous-driving vehicle due to its ability to provide better perception in all light conditions in comparison to other sensors like digital cameras. Adverse weather conditions like fog and rain could reduce the accuracy of the data, but in moderate weather conditions, LiDAR is well suited for high-frequency applications such as building a perception layer for an autonomous vehicle. High-end LiDARs could generate detailed local maps of an ego vehicle working in all light conditions. These maps could be used for a variety of critical tasks such as behavior predictions of the surrounding vehicles and environment. This environmental behavior prediction, such as whether a vehicle ahead is making a turn or not, helps a self-driving vehicle in predictive path planning. Typically, LiDARs are used in medium-range {80 to 160 m} applications such as collision avoidance and pedestrian detection and also in long-range {160 to 300 m} applications like adaptive cruise control and critical object tracking. A smart LiDAR is equipped with integrated ECUs to perform pre-processing, object-recognition (detection and classification), and track-

ing functions and provides a list of tracked objects to a control system. On the other hand, a simple LiDAR provides a raw point cloud, and the object recognition and tracking are performed in the ADAS ECU, as shown in Figure 1.2. The choice of the type of LiDAR depends on autonomous vehicle architecture and functional safety requirements. For this research, we focus on autonomous vehicle systems built on LiDARs that provide a raw point cloud.

The LiDAR data returns have no shape attributes, as they represent the perceived environment. The density of the point cloud depends on the horizontal and vertical angular resolution of the LiDAR. For automotive applications in general, the point cloud is sparse, as the points are spread across the maximum range of the LiDAR, which could be up to 300 m. Each point in the point cloud is usually represented by its Cartesian coordinates and the intensity of reflection. In autonomous vehicle applications, most of the existing object detection and tracking models do not consider the intensity of reflection; hence, that value is neglected. The 3D point cloud is considered as a set of points $pc = \{p^1, p^2, p^3..p^n\}$, where each point is the combination of its x, y, z components $p^i = \{p^i_x, p^i_y, p^i_z\}$.

5.2.1 QIM-based Data Hiding on LiDAR Point Cloud

Any given sample in a LiDAR dataset is the combination of the reflection intensity of a point in space and its corresponding 3D location coordinates. Since we are focusing on autonomous vehicle applications, the primary usage of LiDAR data would be in the areas of perception and localization. These applications require distance measurements to the detected objects in the LiDAR point cloud. In performing object detection on the raw LiDAR point cloud, the general norm is to reduce the redundancy and bring in fixed connectivity between the points and then feed this sensor data to a prediction model. Most of these prediction models are deep-learning-based, where the model extracts features based on the training data set. The existing prediction

models cannot detect LiDAR point-cloud tampering. Cryptographic- or data hiding-based approaches can be developed to solve this problem. It can be observed from Figure 1.4 that the data hiding-based solution outperforms the cryptographic-based solution as far as latency is concerned. The challenge for the data hiding-based integrity verification method for automotive and robotic applications is to ensure that message embedding distortion should not deteriorate the performance of prediction models used in the ADAS unit. The QIM-based data hiding provides the flexibility to select a desired embedding distortion level as a function of the quantization parameter, which is the main motivation behind selecting QIM over other available data-hiding methods. In the following, we outline QIM-based data hiding for a LiDAR point cloud.

The basic principle of quantizing the host signal using multiple quantizers, where each one of them could be treated as a set of reconstruction points, can be extended to a 3D point cloud such as LiDAR sensor data (Joachim and Bernd, 2002). The point samples from the LiDAR sensor are randomly located by default and lack connectivity information. To give them shape and connectivity aspects, the point cloud is divided into fixed-size voxels. A voxel is a fixed-width cube in 3D space. Once the maximum range of the point samples from the sensor is determined, points are quantized with a specific step size Δ. After this quantization step, all the points that fall within a voxel are represented either by a fixed vertex of the corresponding voxel or by its centroid. This voxelization step also reduces the redundancy in reflections from the same target.

After voxelization, all the points of the signal S, with position vectors that fall within a voxel k are represented by the vertex at the origin of the voxel $v_k = \{x_k, y_k, z_k\}$ that assumes a value given by a uniform scalar quantizer $Q(v_k, \Delta)$

$$Q(v_k, \Delta) = round(\frac{v_k}{\Delta}) \cdot \Delta \tag{5.1}$$

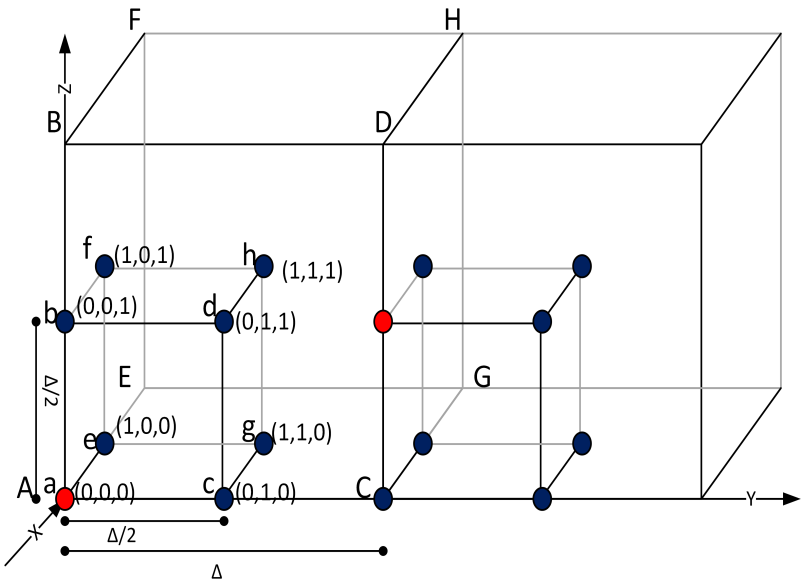


Figure 5.1: Illustration of 3D QIM-based data hiding, here axis representation is in LiDAR frame

Extending the QIM based data-hiding method from section 3.4 to a 3D point cloud gives the ability to embed multiple bits in each point. We implement a simple form of the lattice QIM using the 3D cubic base lattice. Data is hidden in the spatial domain by modifying the three-dimensional position vector of each point; hence, we have three degrees of freedom in the selection of reconstruction points as shown in Figure 5.1. The non-intersecting nature of the reconstruction points results in host-signal interference rejection (Chen and Wornell, 1998). Based on the hidden message tuple, m_k, to be embedded, here $m_k = \{m_{xk}, m_{yk}, m_{zk}\}$ the vertex can be moved around a fixed inner cube of a given dither size. The resulting watermarked signal for 3D QIM can be represented as:

$$s_w(s_{v_k}, m_k) = q_{m_k}(s_{v_k}, \Delta) \tag{5.2}$$

where $q_{m_k}(\cdot)$, denotes 3D QIM quantizer which is expressed as:

$$q_{m_k}(v_k, \Delta) = round(\frac{v_k}{\Delta}) \cdot \Delta \pm \frac{\Delta}{2} \cdot \sqrt{\begin{bmatrix} m_{xk} \\ m_{yk} \\ m_{zk} \end{bmatrix}} \tag{5.3}$$

If we embedded three bits per host-signal sample to take advantage of the three-dimensional spread of points, with the embedding rate $R = 3\ bits/sample$, the embedded message m_k would assume 2^R values. The range of m_k determines the count of the ensemble of quantizers hence the quantizer ensemble will have eight values $q_i \in \{q_1, q_2, q_3, \cdots, q_8\}$ in this case. Each one of these eight quantizers shifts the vertex point at A in Figure 5.1 to one of the eight vertices $\{a, b, c, d, e, f, g, h\}$ within the inner cube. If, for example, all the points of a 3D point cloud are arranged in sequential order, Figure 5.1 represents the first two voxels of the point cloud. If the point cloud is quantized with a step-size Δ, the points within these first two voxels are represented by vertices A and C. In the proposed 3D QIM method, the position of the vertex is moved within an inner hypercube of size $\Delta/2$ based on the embedded message, which is the sequence number of the voxel, i.e., 0 or 1. This shift in the vertex position is depicted by the red circle in Figure 5.1. The proposed method of moving the vertex within an inner hypercube does not increase the vertex count in comparison to a normal quantization and hence does not introduce any additional transmission overhead.

5.3 Attack Modeling

Attacks on LiDAR sensors used in autonomous vehicle applications such as localization and perception can be broadly divided into two categories:

1. Regular-channel attacks at sensor level: Sensor saturation, spoofing.

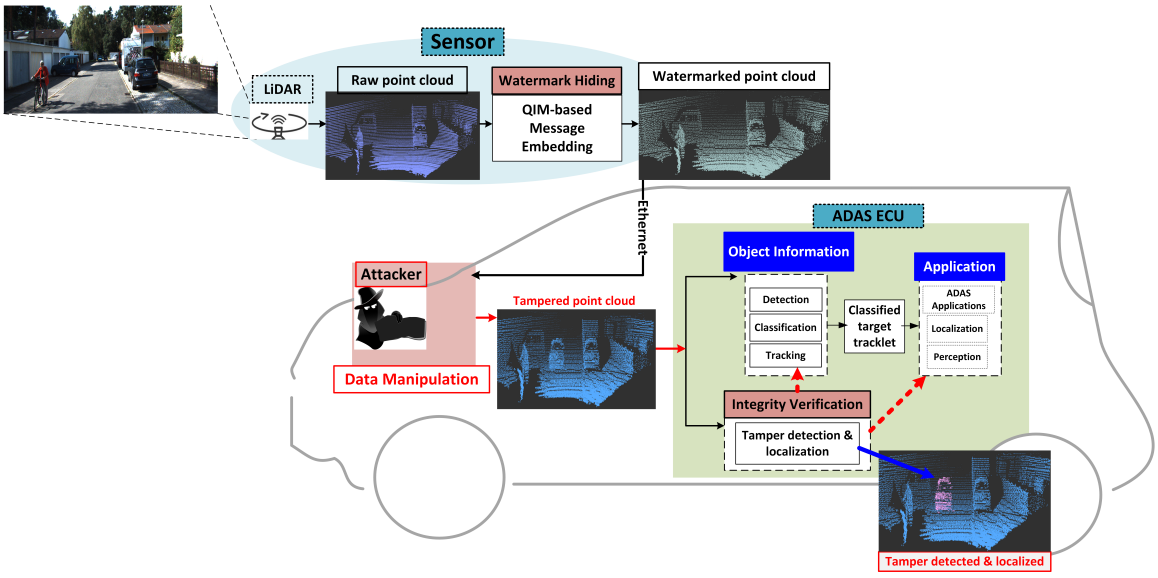


Figure 5.2: Block diagram of the proposed QIM-based framework

2. Transmission-channel attacks at interface level: Point cloud tampering or deformation.

Regular-channel attacks such as sensor saturation (flooding the target with bright light) and relay and replay attacks (capturing and re-sending the target LiDAR pulse sequence) can be launched external to the vehicle but need precise knowledge of the target LiDAR pulse sequence, receiving angles, and listening time interval (Shin et al., 2017), (Petit et al., 2015). These attacks could be nullified by introducing some pre-processing steps like random probing, correlation, and voting-based confidence estimators. The proposed data hiding-based method is unable to detect regular-channel attacks.

For transmission-channel attacks, which can be launched from inside the vehicle, creating a fake scene could be as simple as copying or deleting a section of the point cloud at the desired location. These insider attacks can be launched with ease in real-time and can have a maximum impact on vehicle decision making if the ADAS core algorithms are designed on the assumption that the sensor data is credible. Most

of the object-detection and classification algorithms in the data analysis pipeline are deep-learning-based and are run or inferenced in real-time. These deep-learning models do not differentiate between a fake object and a real object, which could result in erroneous object detections on tampered data. In this section, we describe the attack model for transmission-channel attacks. Transmission-channel attacks happen at the edge system when a hacker gets access to the network interfaces or the decision-making control unit. A hacker could modify the data or point cloud in real-time by some simple operations like copying the existing targets from the point cloud and pasting them in the direct path of the vehicle, which could prompt the vehicle to come to a sudden halt. If the point-cloud tampering is not detected before the inference engine runs on the raw data, it will put more burden on the decision-making logic (the ADAS unit) as it has to incorporate more checks and balances, thus increasing the processing time. Moreover, the ADAS outputs are also expected to be wrong. If we could detect and localize tampering in real-time, then that would ensure the integrity of the sensor data and therefore guarantee the expected ADAS performance.

5.3.1 Attack Vectors

Similar to the attack vectors discussed in Section 4.4, for LiDAR data We have identified two main attack vectors for transmission-channel attacks, which require not only the detection of tampering but also the specific location of the tampering to neglect that area in the decision-making process.

1. **Fake Object Insertion** (FOI): A fake target is inserted in the direct path of the vehicle.

2. **Target Object Deletion** (TOD): An existing target in the path of the vehicle is removed.

5.4 Countermeasures to Transmission Channel Attacks

To counter the above-mentioned attack scenarios on the transmission channel between the LiDAR sensor and the ADAS, we propose a QIM-based data-hiding method for tamper detection and localization. Shown in Figure 5.2 is the block diagram of the proposed method. We divide the framework into an information-hiding processing block at the LiDAR sensor unit and a point-cloud verification and tamper-detection and localization processing block in the ADAS unit. The information-hiding processing framework that is implemented inside the sensor embeds a binary watermark in the raw point cloud, introducing a negligible distortion. After this step, the embedded point cloud is transmitted over the vehicle network. The watermarked point cloud can be directly worked on by the ADAS core to detect and track objects. The integrity-verification processing block runs in parallel to perform integrity checks on the point cloud data in real-time and inputs its decision to the ADAS unit. This verification-processing block localizes the tampered region once it determines that the point cloud is tampered. This approach secures the point cloud against any transmission-channel-intrusion insider attacks, which are hard to detect at the data inference stage.

5.4.1 Implementation Details

In the QIM-based information-hiding processing framework, which runs in close vicinity to the physical sensor, the LiDAR point cloud is filtered to capture forward-looking points or the front camera view. This step can be skipped if the surround-view point cloud is required by the application. The resulting points are quantized based on the predefined step size Δ. After basic quantization, each voxel vertex is shifted to one of the eight positions at a minimum distance of $\Delta/2$ based on a binary message vector as discussed in section 5.2.1. For simplicity in this study, a repeating message sequence $\in \{0, 1, 2, 3, 4, 5, 6, 7\}$ was chosen. At each sample with index i the message

value is given by $m = i \ (mod \ (8))$. The computational complexity of the implemented algorithm is $O(3N)$ for N samples, as embedding each bit per sample is $O(1)$.

For the point-cloud verification framework, a blind watermark extraction mechanism is used, that is, the original point cloud is not used for the watermark extraction process. In the verification block, the embedded message is extracted by quantizing the received signal with the same step-size $\Delta/2$ and selecting the nearest reconstruction point. For simplicity, we have assumed that the verification block is aware of the repeating message-embedding pattern. If a dynamic message embedding is needed, the required pattern can be communicated to the verification block at the receiver end through any selected vehicle interface, as shown in the Figure 5.2.

Based on the correlation values and pattern matching between the embedded and extracted messages, the indices of the received signal where the embedded and extracted messages do not match are determined. From these indices, the corresponding LiDAR frame points are traced and localized as tampered. To measure the accuracy of tamper localization, the Hausdorff distance (Agarwal and Prabhakaran, 2009) is computed between the bounding boxes of the points detected as tampered against the bounding boxes of the ground truth data points.

5.4.2 Performance Evaluation

To evaluate the performance of the proposed framework, we used KITTI vision benchmark data collected using a 64-channel Velodyne HDL-64E LiDAR running at 10 Hz (Geiger et al., 2012). The KITTI offers a sensor synchronized and labeled dataset with the location information of the objects in the data frame. The majority of deep-learning-based object detection and classification models in automated driving domain rely on this dataset for training and performance benchmarking. We also chose this dataset to evaluate the performance of the proposed integrity verification and tamper-resistant methods to keep the analysis as close as possible to

real-world autonomous driving scenarios.

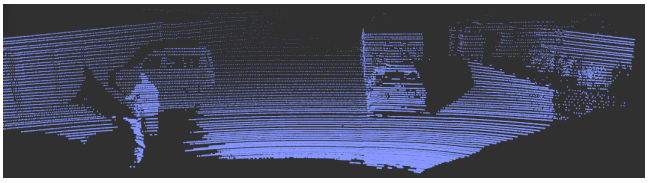
A fake object insertion (FOI) is simulated on the fly by copying a real target object's points from the LiDAR frame at a different location. Similarly, the other attack vector of target object deletion (TOD) is also simulated by removing points in the frame that represents a real labeled target object. These two techniques could be combined to move the targets to a different location, which could be considered as another attack vector. A sample representation of the LiDAR data from the KITTI dataset is shown in Figure 5.3. It visualizes the fake object insertion and known target deletion along with accurate detection and localization. To further understand the effects of channel noise, random Gaussian noise of varying variance is added globally to the LiDAR frame.

The minimum distance d_{min} between two reconstruction points measures the size of the noise vector that can be tolerated by the system (Chen and Wornell, 1998). If we set a limit on message embedding distortion by choosing a fixed quantization step size Δ or, in other words, set a constraint that the composite signal in Equation (5.6) should be closely equal to the s_{v_k} $\forall m_k$, then the message detection accuracy would be high when channel noise is bounded by Equation (4.7). Our experimental results confirmed that the proposed algorithm adheres to this theoretical noise limit.

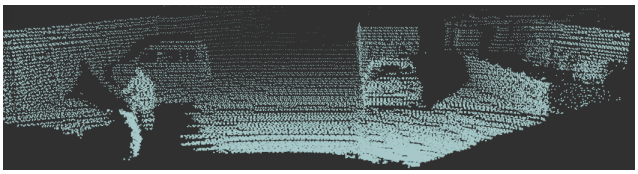
5.5 Experimental Results

The performance of the proposed framework is evaluated on KITTI's 3D object detection benchmark training dataset. For performance evaluation, LiDAR frames with cars in close proximity to the ego vehicle with no occlusions and less truncation are selected. The motivation behind this selection criteria is to aid the visual inspection of the simulated FOI and TOD attack vectors and to precisely evaluate the tamper localization accuracy in a controlled environment. To this end, the first 1000 frames of KITTI's dataset are analyzed that resulted in 67 frames satisfying selection

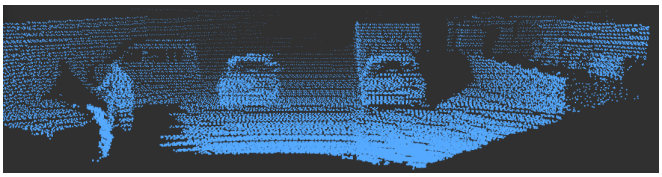
(a) Camera view of LiDAR frame



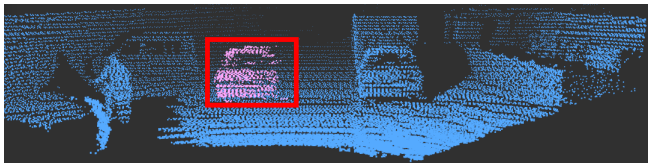
(b) LiDAR frame clean



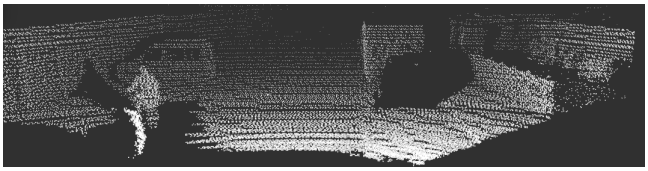
(c) QIM-modulated LiDAR frame



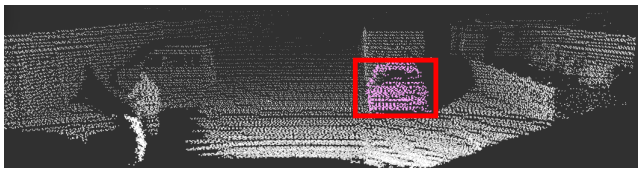
(d) Tamper: Fake car added



(e) Detection: Fake car located



(f) Tamper: Car removed



(g) Detection: Deleted car located

Figure 5.3: Attack models and tamper detection and localization results

criteria. It is important to highlight that the proposed framework applies to all the LiDAR frames in the KITTI's dataset and the LiDAR frame selection criteria is not the limitation of the proposed system. It is rather used to have a more meaningful and fair performance analysis.

The performance of the proposed method is evaluated using four experiments, ranging from investigating the impact on the object detection performance of the ADAS unit of embedding induced distortion to embedding strength analysis and robustness in the presence of additive noise.

5.5.1 Impact of Embedding Distortion on ADAS Performance

The primary goal of this experiment is to investigate the impact of embedding distortion on ADAS functionality. Specifically, this experiment studies the impact of QIM-based data hiding distortion on the performance of object detection and -recognition algorithms. The motivation behind using an object detection algorithm as a key performance indicator in this experiment is because it provides a direct error measurement in terms of the distance between object(s) in the original point cloud and corresponding object(s) in the watermarked point cloud. In other LiDAR applications such as simultaneous localization and mapping (SLAM) and object tracking the message embedding distortion is estimated through indirect methods. In these methods, distortion is estimated as sensor bias and often gets compensated for or canceled based on the filters used in SLAM (Perera et al., 2003).

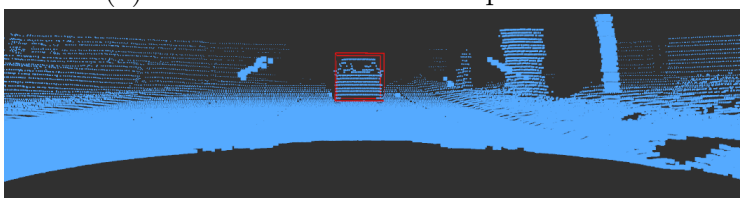
To verify the effect of embedding induced distortion on object detection, as a first step, we ran inference on a selected KITTI dataset frame processed using 3D QIM with nine different step sizes using a pre-trained 3D FCN deep-learning model (Li, 2017). We used an existing implementation of the 3D FCN which was pre-trained on raw KITTI data frames for this experiment (Tsuji, 2018). From the KITTI training data set, we selected a frame in which a target vehicle is within a 50 m

range with zero heading angle. The selected frame is processed using 3D-QIM with nine different step-sizes $\Delta \in \{1, 4, 6, 8, 10, 30, 40, 50\ cm\}$. The resulting frames are run individually through the 3D FCN deep-learning model inference engine, and the resulting bounding box prediction is compared with the ground-truth bounding box. The deviation in terms of the Hausdorff distance between the ground truth and the predicted bounding boxes is compared. The results depicted in Table 5.1, show that the inferencing of the deep-learning model resulted in good accuracy for a step-size of 30 *cm* and below. It can be observed from Figure 5.4 that the shape of the raw point cloud shown in Figure 5.4 (a) with a distinguishable car in the red bounding box (ground truth) deteriorates as we increase the step size Δ. It can be observed that the green bounding box corresponding to the model prediction starts moving away from the red box corresponding to the ground truth as the step-size increases beyond 30 *cm*. As we move farther, with the 64-channel LiDAR data, the number of points representing a target becomes much smaller and falls into single digits. For those labels, we observed a variation in prediction from the raw frame to the QIM-modulated frame even at a smaller $\Delta = 5\ cm$. Since the probability of false alarms is high for objects with low reflection points, they are generally filtered by the decision-making process. The range at which this filtering occurs depends on the LiDAR resolution.

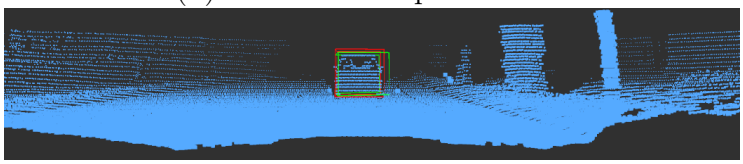
StepSize Δ (cm)	Bounding box shape distortion (m)
1	0.44
4	0.78
6	0.73
8	0.78
10	0.73
30	0.90
35	17.36
40	22.70
50	28.08

Table 5.1: QIM-induced distortion at different step-sizes

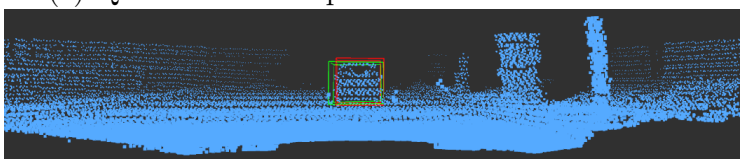
(a) Camera view of LiDAR point cloud



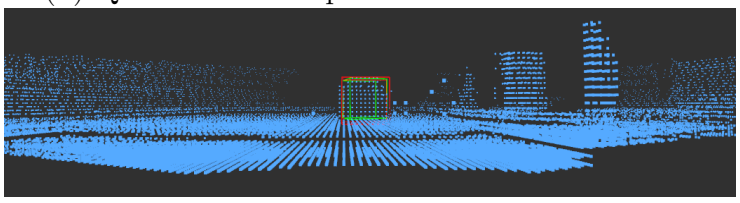
(b) LiDAR: Raw point cloud



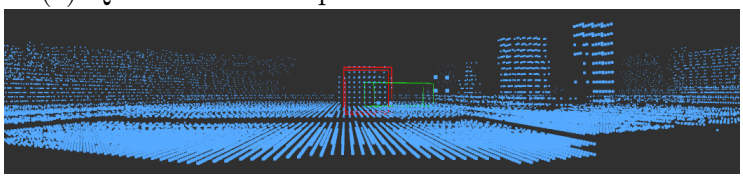
(c) QIM-modulated point cloud with $\Delta = 5$ *cm*



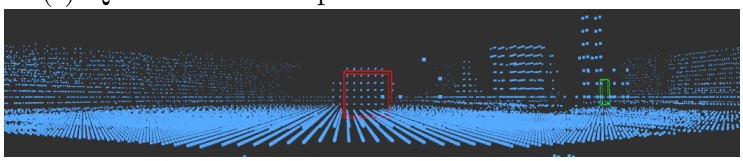
(d) QIM-modulated point cloud with $\Delta = 8$ *cm*



(e) QIM-modulated point cloud with $\Delta = 30$ *cm*



(f) QIM-modulated point cloud with $\Delta = 35$ *cm*



(g) QIM-modulated point cloud with $\Delta = 50$ *cm*

Figure 5.4: Bounding box estimation of a ground truth label at different QIM-embedding step sizes

Δ/Step size (cm)	Bird's Eye View	3D Detection
0	96.92	77.38
1	96.59	83.05
4	96.96	73.12
6	96.21	72.90
8	97.61	75.18
10	89.40	65.30
20	83.50	57.39
30	73.42	34.65
35	54.77	23.51
40	42.01	13.40
50	12.20	2.38

Table 5.2: VoxelNet: Car detection average precision scores

To further understand the effect of message embedding induced distortion on LiDAR object detection accuracy, we tested watermarked LiDAR frames on another 3D object detection model called VoxelNet (Zhou and Tuzel, 2017). VoxelNet is an end to end deep learning network that stacks the voxelization, convolution, and region proposal network (RPN) operations to detect and localize objects from the raw 3D LiDAR point cloud and its performance is claimed to be better than 3D FCN (Zhou and Tuzel, 2017). The VoxelNet implementation determines object detection precision based on the 70% overlap of the predicted 3D and 2D (bird's eye view) bounding boxes with their corresponding ground truth. In this experiment, we used an existing implementation of the VoxelNet that is trained on KITTI benchmark data to detect cars (Huang, 2018). We chose a validation set of 25 frames that fall under the easy detection category defined by KITTI and ran inference on them using a pre-trained model checkpoint. A base-line average precision score of VoxelNet is established by running inference on the selected validation set multiple times. Same set of 25 LiDAR frames watermarked using 3D-QIM with different step-sizes $\Delta \in \{1, 4, 6, 8, 10, 20, 30, 40, 50\ cm\}$ are then generated. VoxelNet inference is executed on each watermarked dataset to get the average precision score as shown in Table 5.2. The first row ($\Delta = 0$) of Table 5.2 shows the average precision of the model for raw

data frames. For the raw data frames, the average precision of 2D and 3D detections was 96.92% and 77.38% respectively.

It can be observed from Table 5.2 that for watermarked frames there is no significant deterioration in the average precision of the model for up-to a step size of 8 cm. Within this range, the bird's eye view detection scores had a mean of 96.85% with a 0.51% standard deviation. The 3D detection scores averaged at 76.32% with a standard deviation of 4.17%. The additional spread in 3D prediction scores in comparison to the 2D scores could be attributed to the fact that the 3D detection is a more challenging task as it requires more accurate localization of shapes in 3D space (Chen et al., 2017) and hence model needs to be trained on a larger data-set to be able to generalize well. It can also be observed from Table 5.2 that for both methods, the average precision score of the VoxelNet model decreases significantly as the step size goes above 8 cm. Modifying the voxel dimensions of the model and training the model on the modulated point cloud could improve the model performance in general. Nevertheless, since our proposed data-hiding technique provides flexibility to select the desired embedding distortion level as a function of the step-size, for any given application an optimum step size could be selected based on the empirical evaluation of the model and application needs.

5.5.2 Embedding Distortion Analysis

This experiment is designed to investigate the impact of single- vs multiple-bit message embedding on tamper localization accuracy. Specifically, for a given step size $\Delta = 10$ cm, we compared bounding box prediction results of the following three QIM message embedding methods under various added noise levels:

1. $1D$ QIM with one-bit embedding along the x-axis

2. $2D$ QIM with two-bit embedding along x,y axes

3. $3D$ QIM with three-bit embedding along x,y,z axes

Shown in Figure 5.5 is the performance of different bit-embedding methods in localizing the tampered area in a point cloud forged with FOI and global uniform noise.

It is observed that the tampered point localization accuracy decreased as the additional uniform noise levels increased. For a given step size $\Delta = 10\ cm$, the tamper localization accuracy of $1D$ QIM is good up to an additional uniform noise of $\sigma = 2.5\ cm$, $2D$ QIM is good up to $\sigma = 2.4\ cm$, and $3D$ QIM is good up to $\sigma = 1.4\ cm$. The noise tolerance values for a given Δ are within the limits of the σ values for $N = 1$, 2, *and* 3 as per Equation (4.7). Figure 5.6 shows the performance of different bit-embedding QIM methods in the presence of additional Gaussian noise. It is observed that there is no significant difference in accuracy between multi- and single-bit embedding.

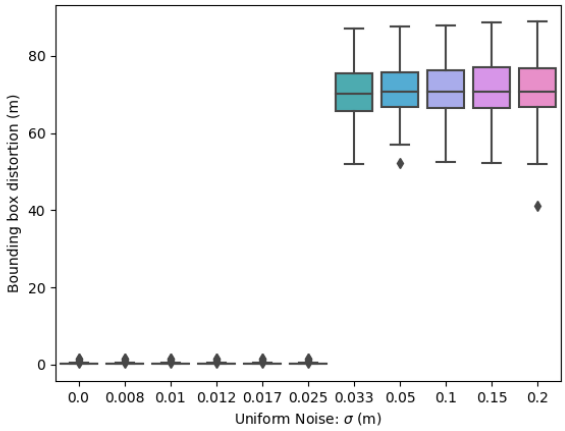
One of the goals of the data hiding is to detect tampering under high lossy conditions such as compression. The data-hiding method should detect any global intentional attacks on the integrity of data such as sensor saturation by external noise addition or affine transforms in multiple dimensions, along with local attacks like FOI and TOD. Though single-bit embedding offers higher robustness to noise levels, it will not detect targeted attacks in multiple dimensions. Hence, we propose using $3D$ QIM for autonomous vehicle applications.

5.5.3 Robustness Analysis

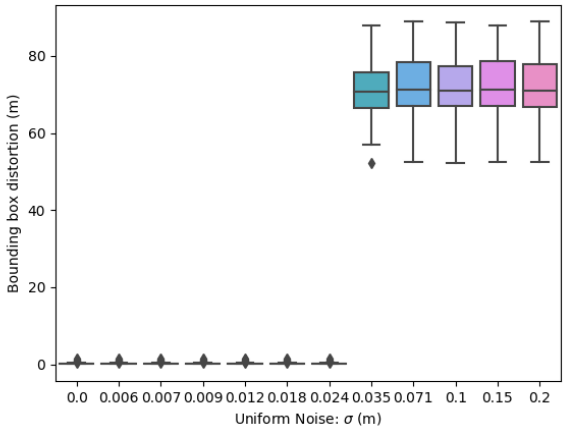
After choosing a range of step-sizes that result in acceptable embedding-induced distortion to analyze the tamper detection and localization accuracy of the proposed 3D QIM method, three attributes are considered:

1. Bit error rate (BER) of embedding

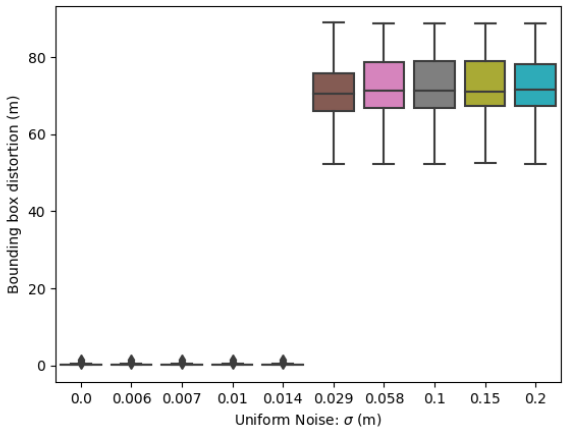
2. Tamper localization distortion



(a) One-bit embedding

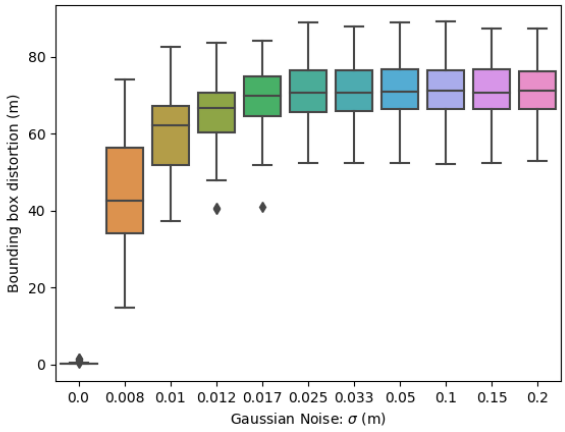


(b) Two-bit embedding

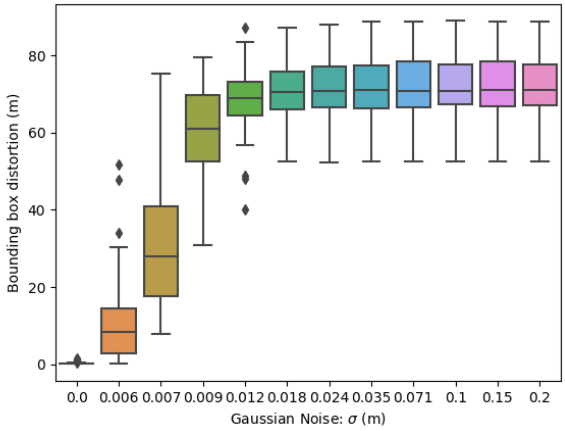


(c) Three-bit embedding

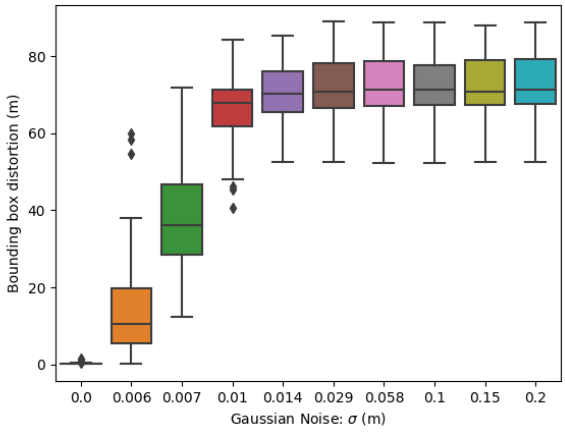
Figure 5.5: Bounding box distortion analysis for different bit-embedding schemes under Uniform additive noise attack



(a) One-bit embedding



(b) Two-bit embedding



(c) Three-bit embedding

Figure 5.6: Bounding box distortion analysis for different bit-embedding schemes under Gaussian additive noise attack

3. Tamper detection false-alarm rate f_{ar}

In different experiments, these three attributes were measured under various variance levels of both uniform and Gaussian noise addition, along with analysis of the TOD and FOI attack vectors and results.

5.5.3.1 Bit Error Rate

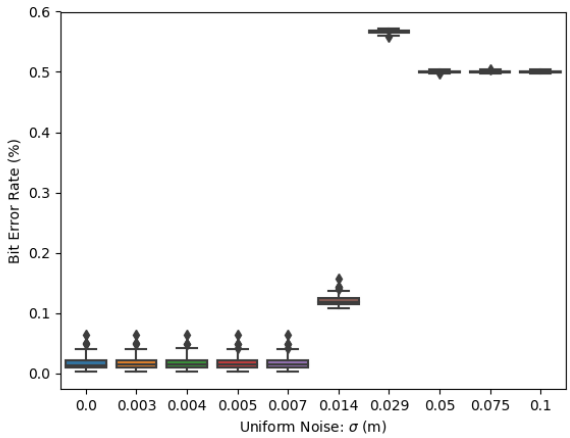
This experiment is designed to measure the performance of the proposed method in terms of bit error rate (BER) in the presence of additive channel-noise. The output of the message-decoding step in our method is a decoded message bit stream from the received signal $\hat{M} = \{\hat{m}_1, \hat{m}_2, \cdots, \hat{m}_N\}$. In this experiment, the bit error rate is measured for each LiDAR frame after the decoding step, where each extracted bit from the received LiDAR frame, $\hat{m}_i$, is compared with the embedded message bit as shown in Equation (4.22). To achieve this goal, Gaussian noise with different standard deviations and uniform noise with different upper bounds are added into watermarked point-cloud frames separately, and the impact of the additive noise attack on BER performance is evaluated. For a clean QIM-modulated frame, the BER is close to 0%. As we tamper with the data by the attack vectors FOI and TOD, a constant BER of close to 2% is observed at zero added noise. As the added noise value increases, the proposed three-bit QIM quantization method maintains a bit error rate of less than 2%, for an added uniform noise of $\sigma < \Delta/6.93$). This noise threshold as defined by Equation (4.7) is $\{0.7, 2.9, 5.1\ cm\}$, respectively, for step-sizes of $\Delta \in \{5, 20, 35\ cm\}$. It can be observed from Figure 5.7 that the BER values are within acceptable bounds until the noise levels exceed the threshold defined by Equation (4.7). The BER values go high at levels below the threshold bounds defined by Equation (4.7) for added Gaussian white noise, as observed in Figure 5.8. This change can be attributed to the 32% noise values falling outside the 1σ range in the Gaussian distribution. Figures. 5.7, and 5.8 show the BER for one of the

attack vectors, FOI. A similar trend is observed for the TOD attack vector.

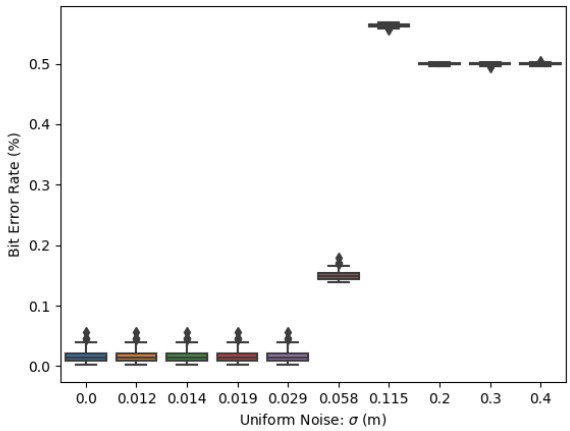
5.5.3.2 Tamper Detection and Localization

This experiment aims to measure the accuracy of the tamper localization feature of the proposed method and how the accuracy is affected by added channel noise. As part of the decoding step, the indices of the tampered points are extracted from the received frame. A bounding box enclosing these points is generated, and the corners of this bounding box are compared with the corners of the ground-truth bounding box provided by KITTI to get a measure of their proximity. We calculated the Hausdorff distance between the two corner sets to measure the maximum distance of a given vertex from the ground-truth bounding box to a similar vertex in the predicted bounding box. Smaller values of this localization-distortion attribute suggest higher accuracy of the prediction or, in other words, suggest that the proposed method can draw a boundary across the tampered points accurately. We added Gaussian noise with different standard deviations and uniform noise with different upper bounds to the QIM-modulated frames and measured the effect of added noise on the performance of the localization accuracy. The added noise is tested at varying levels of σ in the range $\sigma \in \{0.0, \Delta/(10\sqrt(N), \Delta/(8\sqrt(N)), \Delta/(6\sqrt(N)), ..., 2\Delta\}$ and step-size Δ in range $\Delta \in \{5, 10, 20, 30, 35, 40\ cm\}$, where $N = 3$ is the number of bits used for embedding.

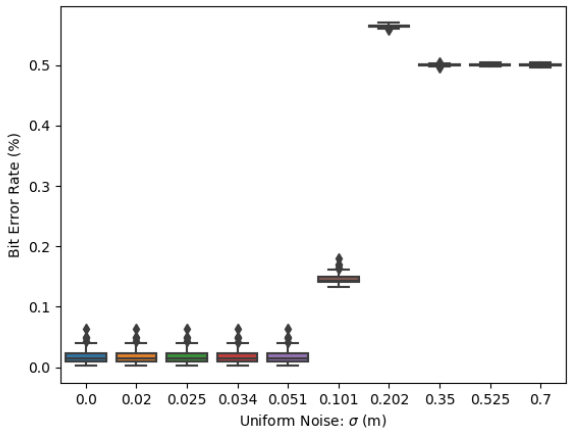
Shown in Figure 5.9 are the localization distortion box plots for the FOI attack vector measured on point clouds with added uniform noise. It is observed that the localization distortion is less than 2 cm as long as the noise level $\sigma < d_{min}/(2*\sqrt(N))$, where $d_{min} = \Delta/2$ and $N = 3$. These results are in sync with the theoretical limits given by Equation (4.7) and demonstrate that the proposed method can localize tampering in the point cloud accurately in the presence of bounded noise. The performance of the proposed method under added Gaussian noise is shown in Figure 5.10.



(a) $\Delta = 5\ cm$

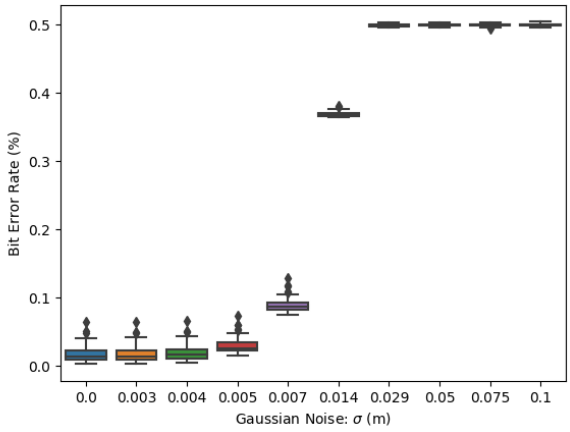


(b) $\Delta = 20\ cm$

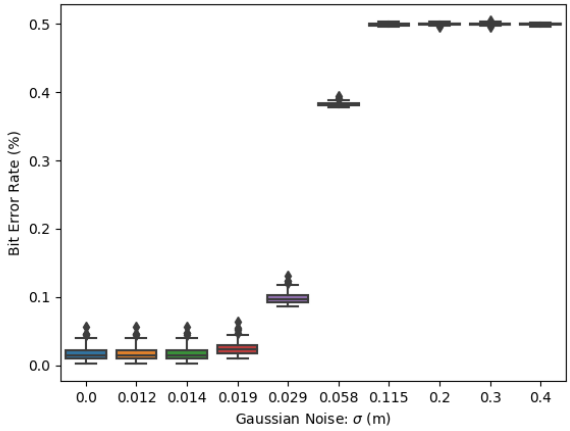


(c) $\Delta = 35\ cm$

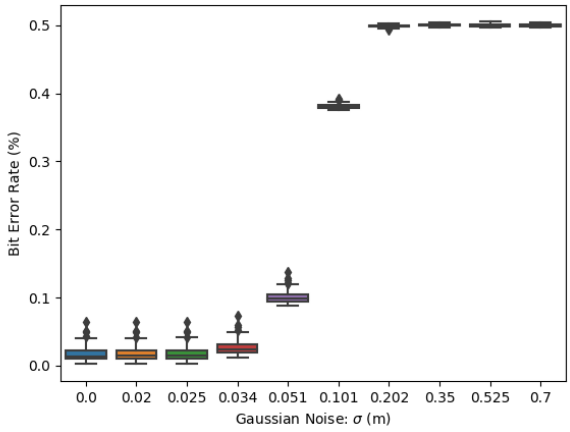
Figure 5.7: Bit error rate of decoded code book for different step sizes and added uniform noise



(a) $\Delta = 5\ cm$



(b) $\Delta = 20\ cm$



(c) $\Delta = 35\ cm$

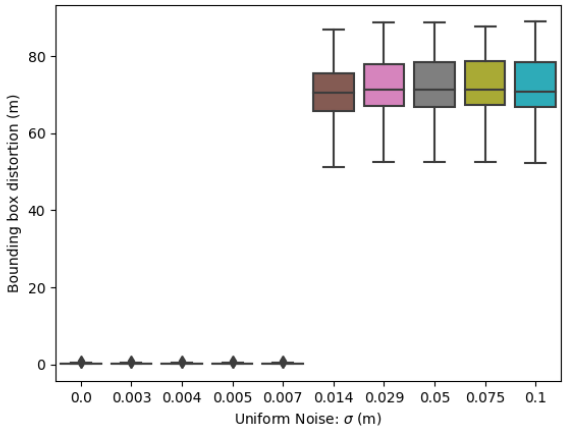
Figure 5.8: Bit error rate of decoded code book for different step sizes and added Gaussian noise

For added Gaussian noise, we observed that the localization distortion is less than 2 cm as long as the noise level $\sigma < d_{min}/(5 * \sqrt{(N)})$, where $d_{min} = \Delta/2$ and $N = 3$. Again, this behavior of low robustness to noise can be attributed to the noise samples that fall outside the 1σ range in Gaussian noise. Similar trends were observed for the TOD attack vector.

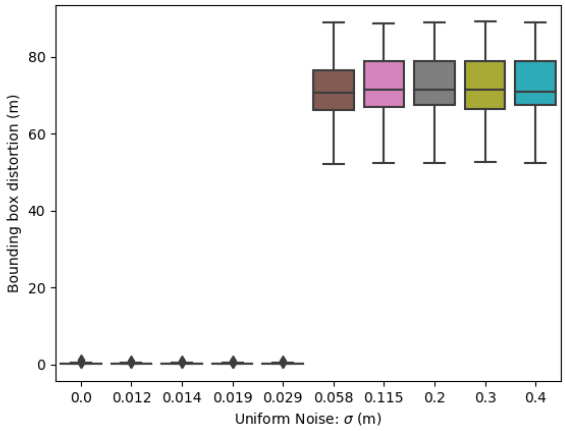
5.5.3.3 False-alarm Rate Analysis

This experiment aims to measure the false-alarm rate of the proposed method in detecting the tampered point cloud in the presence of additive noise as per Equation (4.23). In this experiment, we tracked the number of frames that our algorithm falsely detected as tampered when it was given a clean frame.In this test, the tamper detection false-alarm rate f_{ar} stayed at 0% when there was no added channel noise. In other words, the proposed model detected the presence of both the FOI and TOD attack vectors accurately when there was no added channel noise. When noise was added along with the attack vectors, for added uniform noise, f_{ar} stayed at 0% for $\sigma < d_{min}/(2 * \sqrt{(N)})$, where $d_{min} = \Delta/2$, $N = 3$ and $\Delta \in \{5, 10, 20, 30, 35, 40\ cm\}$.

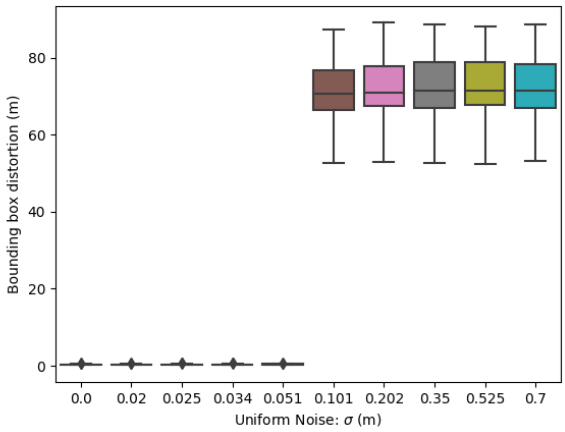
As the noise σ level increased beyond that threshold, the f_{ar} value jumped to 100%, as shown in Table 5.3. It is also observed that the f_{ar} value increased to 100% at lower thresholds of $\sigma > d_{min}/(10 * \sqrt{(N)})$ in the case of Gaussian added noise, and this could be attributed to the noise values grater than 1σ. It can be observed from Table 5.3 that at a given Δ within acceptable distortion bound, the proposed method can achieve 100% accurate detection and localization. A similar trend was observed for the TOD attack vector. In automotive applications, the ethernet and other local networks are not susceptible to high channel noise; hence, our proposed method is expected to achieve the desired performance.



(a) $\Delta = 5\ cm$

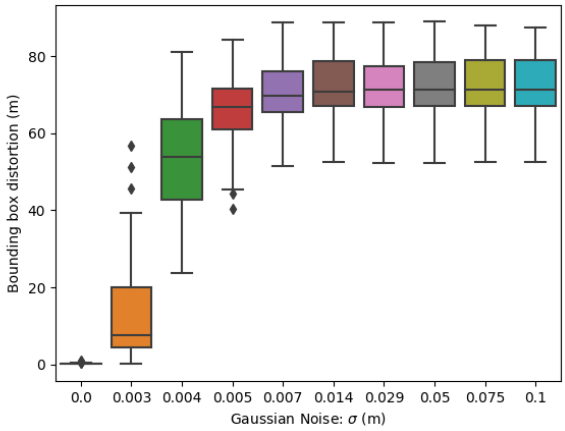


(b) $\Delta = 20\ cm$

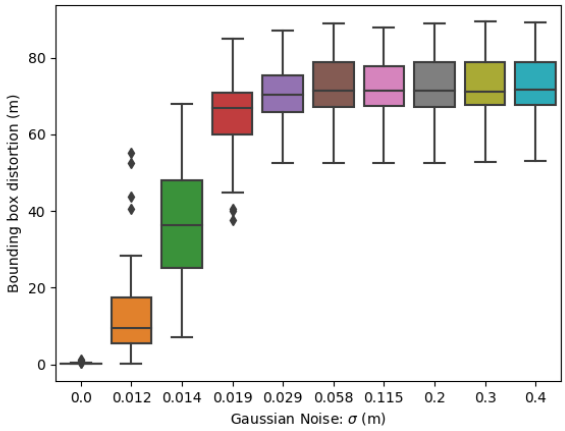


(c) $\Delta = 35\ cm$

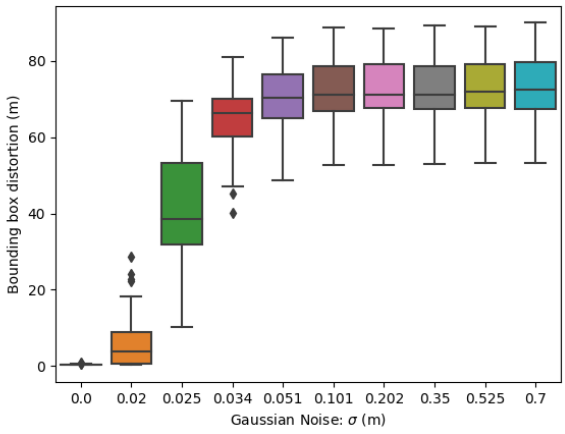
Figure 5.9: Bounding box distortion in meters for different step sizes and added uniform noise



(a) $\Delta = 5\ cm$



(b) $\Delta = 20\ cm$



(c) $\Delta = 35\ cm$

Figure 5.10: Bounding box distortion in meters for different step sizes and added Gaussian noise

5.6 Vulnerability Analysis of QIM

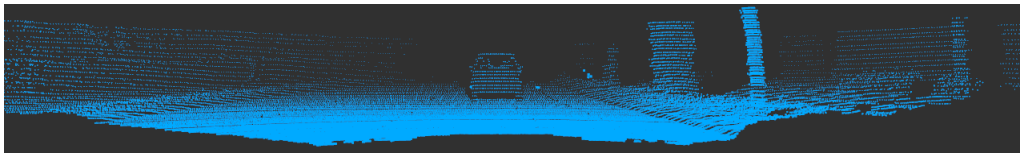
The plain QIM framework proposed above has some security vulnerabilities due to the use of plain QIM for embedding and plain text exchange of the embedded message sequence. The effectiveness of data hiding schemes like QIM is measured by their robustness against the message estimation attack. An estimation attack is a process where the embedded message is estimated through the codebook or step-size (Δ). Accurate estimation of Δ can lead to the estimation of embedded message

$\Delta = 5$	$\sigma(\ cm)$	0	0.3	0.4	0.5	0.7	1.4
	Gaussian	0	0.95	1.0	1.0	1.0	1.0
	uniform	0	0	0	0	0	1.0
$\Delta = 20$	$\sigma(\ cm)$	0	1.2	1.4	1.9	2.9	5.8
	Gaussian	0	0.94	1.0	1.0	1.0	1.0
	uniform	0	0	0	0	0	1.0
$\Delta = 35$	$\sigma(\ cm)$	0	2.0	2.5	3.4	5.1	10.1
	Gaussian	0	0.55	1.0	1.0	1.0	1.0
	uniform	0	0	0	0	0	1.0

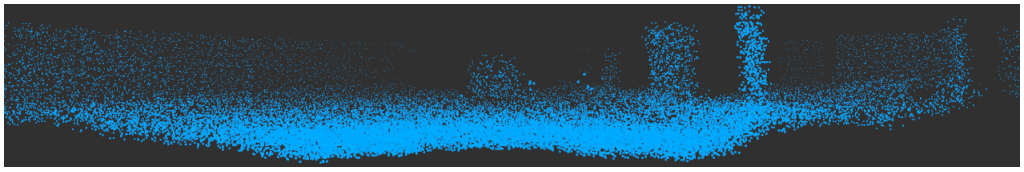
Table 5.3: False alarm rates at different step-sizes for added noise

features (Malik et al., 2008). The plain QIM method is more vulnerable to the estimation attack when compared to dither QIM due to its regularity and predicted outcome. This phenomenon is depicted in Figure 5.11, where-in a sample LiDAR frame is embedded with a given message stream using both dither QIM and plain QIM methods. It can be observed from Figure 5.11, that the plain QIM frame points are regularly spaced where-as the dither QIM frame points display randomness in spacing for a given step-size $\Delta = 35\ cm$. This randomness adds more uncertainty to watermark embedding hence making it more challenging to estimate. A relatively simple step-size estimation algorithm proposed in (Malik et al., 2008) can estimate the Δ of QIM-stego images of varying message lengths and embedding rates with 100% accuracy. A similar approach would fail for the dither QIM stego image due to the randomness in the quantization step-size. A similar pattern is observed even in

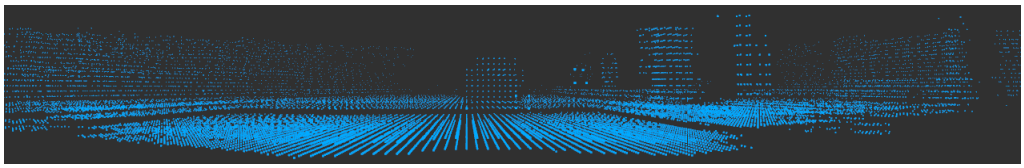
the passive steganalysis approach using non-parametric methods like average entropy estimation (Malik et al., 2008). The images embedded using normal QIM are detected with high accuracy (0.2% false negatives) when compared to the images embedded with dither QIM (7% false negatives).



(a) Clean LiDAR frame



(b) Dither modulated LiDAR frame



(c) Plain QIM modulated LiDAR frame

Figure 5.11: Illustration of quantization noise

Based on this vulnerability analysis, we chose the dither QIM method to enhance the sensor data integrity verification process. In dither modulation based QIM, different factors such as the dither range (DR) and the sample length of embedded message sequence determine the tamper detection and localization accuracies of the framework. We deduce the optimal values of those parameters and discuss the trade-offs between the channel noise tolerance and the tamper detection and localization accuracy using LiDAR sensor data.

5.6.1 Countermeasure Framework

To overcome the security vulnerabilities of the countermeasure framework proposed in section 5.4, we propose a more secured quantization method called spread 3D dither QIM. In this new approach, which is a combination of cryptography and watermarking, the initial handshake between the sensor and the central ADAS unit relies on symmetric or asymmetric cryptography to transmit embedded message sequences and other data hiding parameters. As shown in the sequence diagram of Figure 5.12. After this initial data exchange, the sensor transmits the 3D dither modulated frames to the ADAS unit. The ADAS unit can directly work on these frames as the embedding induced distortion is minimal and does not affect the inference engine. In parallel, the decoding algorithms extract the embedded data to detect and localize the tampered area in the data frame. The data validity flag would indicate further applications to use or ignore that particular data from the sensor.

5.6.1.1 Dither Modulation

In the proposed method based on the dither modulation to embed a binary message $m \in 0, 1$ the message bits 0 and 1 are embedded using the quantizers Q_0 & Q_1 as defined below

$$Q_0(x, \Delta) = \text{round}\left(\frac{x + d_0}{\Delta}\right) \cdot \Delta - d_0 \tag{5.4}$$

$$Q_1(x, \Delta) = \text{round}\left(\frac{x + d_1}{\Delta}\right) \cdot \Delta - d_1 \tag{5.5}$$

in Equation (5.4) & 5.5, dither values d_0 and d_1 are defined by

$$d_1 = \begin{cases} d_0 + \frac{\Delta}{2}, & d_0 < 0 \\ \\ d_0 - \frac{\Delta}{2}, & d_0 > 0 \end{cases} \tag{5.6}$$

In Equation (5.6) the dither value d_0 is pseudo-randomly chosen over a uniform

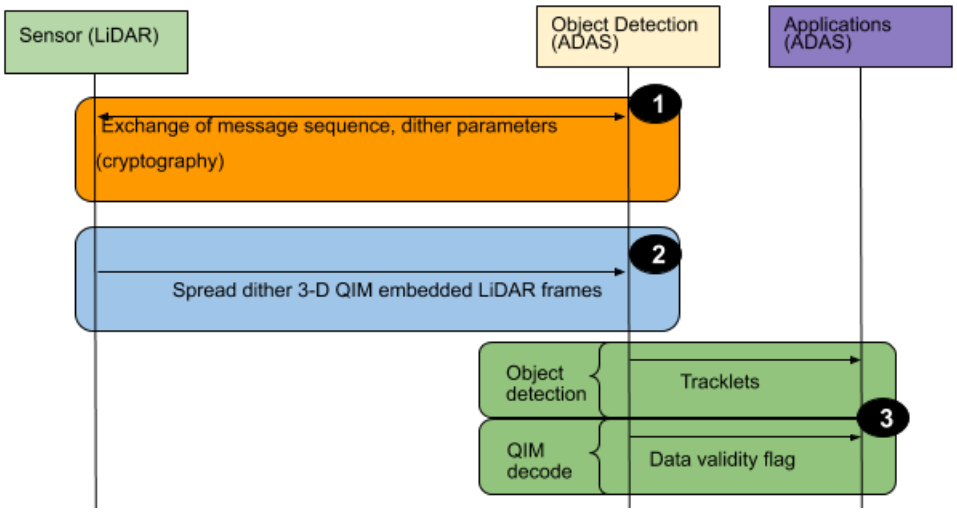


Figure 5.12: Sequence diagram of proposed method

distribution over a min and max range of $[-\frac{\Delta}{2}, \frac{\Delta}{2}]$, for a given step size Δ. With this embedding mechanism, the data hiding technique cannot be easily reverse-engineered even when the embedding message scheme is compromised. The quantized signal value is represented as (Bitar et al., 2015)

$$x_q = x + (Q_m(x, \Delta) - x) \quad m \in \{0, 1\} \tag{5.7}$$

The extraction of embedded message m_d is performed using the minimum distance decoder such as

$$m_d =_{m \in 0,1} |x_q - Q_m(x, \Delta)| \tag{5.8}$$

Applying the dither QIM approach to 3D QIM, the embedding region becomes a hyper-cube, and the set of points obtained by perturbing the center of the hyper-cube to all possible vertices are given by a coset vector (Wenjun et al., 2006). These

coset vectors for a 3D QIM can be represented as $S_0, \cdots, S_7$

$$W_{i,j,k} = \begin{cases} S_0 = 000 & W_i^1 = 0, W_j^1 = 0, W_k^1 = 0 \\ S_1 = 001 & W_i^1 = 0, W_j^1 = 0, W_k^1 = 1 \\ \vdots & \\ S_7 = 111 & W_i^1 = 1, W_j^1 = 1, W_k^1 = 1 \end{cases} \tag{5.9}$$

In Equation (5.9), $W_{i,j,k}$ represents embedded message symbol value at a given 3D voxel centeroid of the point cloud represented by its x, y, z position. In 3D QIM the message symbol can contain up to three bits and the numeric values of message m can be in range $m \in \{0, \ldots, 7\}$. For the voxelized LiDAR data, the dither modulation would move the voxel centeroid based on the embedding message sequence as shown in Figure 5.13 In the spread 3D QIM method, each message symbol is embedded into

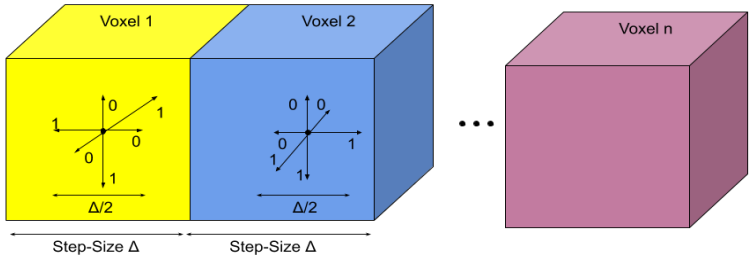


Figure 5.13: Voxel centeroid movement due to dither modulation

the host signal vector for a length of (L) samples. The advantage of this spread is the spread of the distortion across groups of samples that increases the embedded message extraction accuracy.

5.6.1.2 Watermark Generation

The watermark used in this method is a numeric sequence $m \in \{0 \cdots 7\}$ that is generated based on a random seed. The exchange of the watermark message sequence and the dither modulation parameters can happen one time during the sensor initialization on power-up through asymmetric cryptographic encoding. Also, to further

fortify the system, these parameters could be exchanged between the sensor and receiver at random time intervals to reduce further the chances of a man in the middle attack where the hacker learns the watermark sequence.

5.6.1.3 Watermark Embedding

For embedding the hidden message into the LiDAR point cloud using the proposed spread 3D dither QIM, first, the LiDAR data frame is quantized into voxels as explained in section 5.4. For a given frame if LF_{vl} denotes the LiDAR frame voxel length, a message with a symbol length $m_{sl} \leq LF_{vl}$ is selected. This message can be ciphered by applying the XOR operation and a secret key or any other cryptographic algorithm. Each symbol of the cipher message, which is represented by 3 bits, is embedded into a group of L voxels ($L \geq 1$) in the LiDAR frame. In this process, each bit of the three-bit message symbol is embedded into the (x, y, z) coordinates of the voxel centroid, as shown in Equation (5.9).

To embed a message string of length k, into a host signal x with L samples at a time, we need $k \times L$ voxels. If L is chosen to be say six, then to embed three message symbols we need eighteen voxels as shown Figure 5.14. Each bit of the encoded message is inserted into six samples that correspond to the (x, y, z) coordinates of the six samples $(x_0, y_0, z_0), (x_1, y_1, z_1), \cdots, (x_5, y_5, z_5)$. After the embedding process, the six samples become: $(x_0^{'}, y_0^{'}, z_0^{'}), (x_1^{'}, y_1^{'}, z_1^{'}), \cdots, (x_5^{'}, y_5^{'}, z_5^{'})$, with each voxel centeroid moved by a dither value within a hypercube of size $\Delta/2$. The embedded message can be extracted using the Equation (5.8).

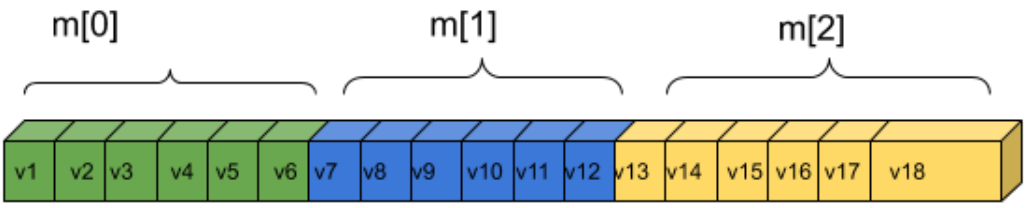


Figure 5.14: Message embedding example with three bits and L= 6

5.6.2 Experiments & Results

As discussed in section 5.5.2, the acceptable levels of watermarking induced distortion in the autonomous vehicle applications, for LiDAR object detection in specific, depends on the detection models used and their corresponding constraints. In QIM based data hiding approach the step-size Δ can be adjusted to produce acceptable levels of distortion based on the choice of the detection algorithm. In the distortion analysis presented in section 5.5.2, the maximum distance between the reconstruction points is $\Delta/2$. Within this range, in the proposed method of 3D spread dither QIM, we have additional parameters such as dither range (DR) and spread sample length (L) that can affect the tamper detection accuracy and noise robustness. To understand the effects of these parameters on the tamper detection and localization accuracy, multiple experiments are performed to deduce values for dither-range and spread length for a given step-size Δ. Experiments focus on three features for the robustness analysis

- Encoding bit-error-rate (BER),
- Tamper localization accuracy, and
- Tamper detection false-negative rate f_n.

These features are computed for various attack vectors in the presence of uniform Gaussian noise. These features are tested at the noise boundary conditions such as $\sigma < DR$ and $\sigma > DR$ to verify the performance of the proposed method. Also to determine the impact of the spread of the dither modulation, multiple spread sample lengths L are used in performing the boundary condition checks.

5.6.2.1 Dataset

The proposed method of spread dither 3D-QIM encoding and decoding is tested on random urban driving scenario LiDAR frames from the KITTI vision benchmark

suite. Building on top of the analysis done in section 5.5.3.2 to choose the appropriate step-size to reduce the QIM embedded distortion, here, we focus on methods to further secure the encoding mechanism and understand the trade-offs in implementing those methods.

5.6.2.2 Bit Error Rate

In this experiment, the BER is measured by comparing the embedded message bitstream $message_{embedd}$ with the decoded message bitstreams $message_{decode}$. The number of mismatches $error_{bits}$ are divided by the length of the bitstream $len(message_{decode})$ to get the rate of error as per Equation (4.22). Varying levels of uniform noise is added to the embedded frames, and BER values are measured at different spread sample lengths (L). It is observed that the BER values increased with the increase in the dither range (DR) values irrespective of the sample length. For a given step size Δ, the BER values are higher when the dither range nears the max values $\pm\Delta/2$. As the dither range values get lower, the BER values improve and get below the 2% range for higher sample length spread. This trend persists even when the added noise variance σ is greater than the QIM toleration theoretical toleration limit shown in Equation (4.7) Figure 5.15-5.17 show the average bit error rate values calculated by the counter framework pipeline at different spread lengths L and dither ranges DR.

5.6.2.3 Localization Accuracy

The LiDAR frames once encoded with the proposed method of spread dither 3D QIM method, are tampered with the attack vectors in this experiment. The accuracy with which the proposed method localizes the tampered area in the tampered LiDAR frames is estimated by calculating the Hausdorff distances between the 3D bounding boxes encompassing the tampered area and the corresponding ground truth bounding

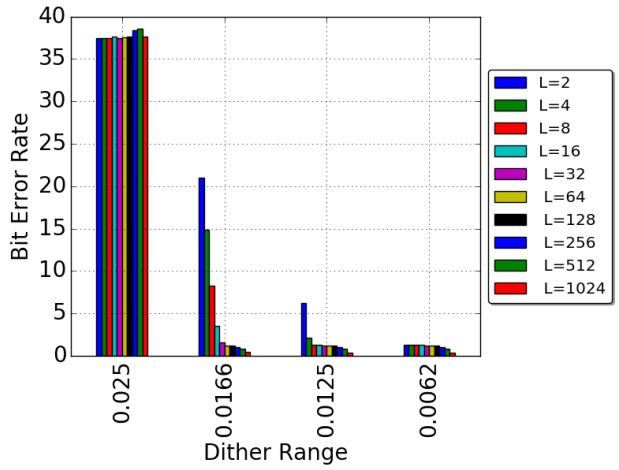


Figure 5.15: Bit Error Rate at $\Delta = 5cm, \sigma = 0.0$

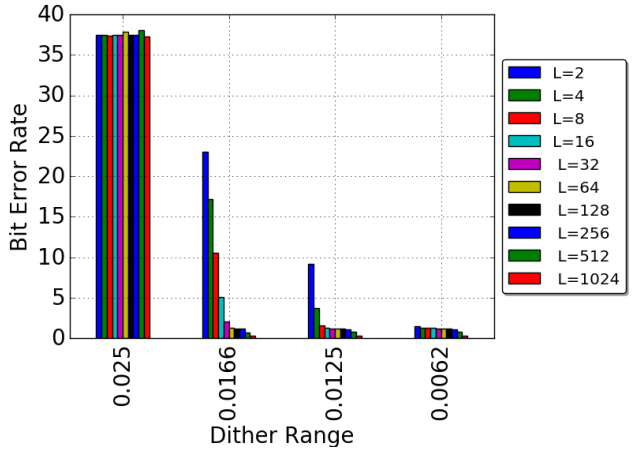


Figure 5.16: Bit Error Rate at $\Delta = 5cm, \sigma = 0.0072$

boxes.

These calculations are repeated for multiple data frames with additional uniform noise for a given step-size $\Delta = 5\ cm$. It is observed that the localization accuracy is in general low for higher dither ranges $\pm\Delta/2$ at all spread lengths. As the dither range decreases, the spread length values in the extremes still show lower accuracy. The optimum spread lengths for accurate localization of tamper are $\{8, 16, 32\}$ at the dither ranges $\{\Delta/4, \Delta/8\}$ for clean LiDAR frames without added noise. Here as the dither range goes above $\Delta/4$, the probability of decoding the wrong voxel vertex increases. Though the spread should decrease this error by increasing the number of

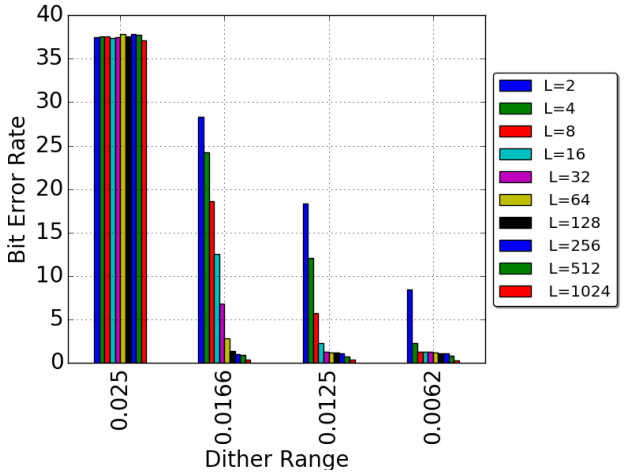


Figure 5.17: Bit Error Rate at $\Delta = 5cm, \sigma = 0.0144$

occurrences, in this particular use case with the tamper points covering 10% of the total frame length which is around $\sim 30K$ voxels, the experiments show that having a spread length of $> 10\%$ of the tamper points results in inaccurate localization of the tampering.

As the noise, σ increases, the acceptable localization accuracy levels are achieved at the lower dither ranges $\Delta/8$ and the sample lengths of $\{16\}$ or $\sim 5\%$ of total sample points seem to be the optimum length for spread. This analysis provides an important insight into the selection of dither ranges and spread lengths for autonomous vehicle applications. The theoretical tolerance for QIM at a given step size is given by Equation (4.7). The dither range that falls within this range is $\{\Delta/8\}$, and even within this range, the optimum length of the spread for the localization accuracy is deduced to be $\sim 5\%$.

Figure 5.18-5.20 show the average localization distortion values calculated by the counter framework pipeline at different spread lengths and dither ranges.

5.6.2.4 False Negatives

In this experiment, the number of tampered LiDAR frames that escaped the detection pipeline is estimated. The two main issues with probabilistic functions

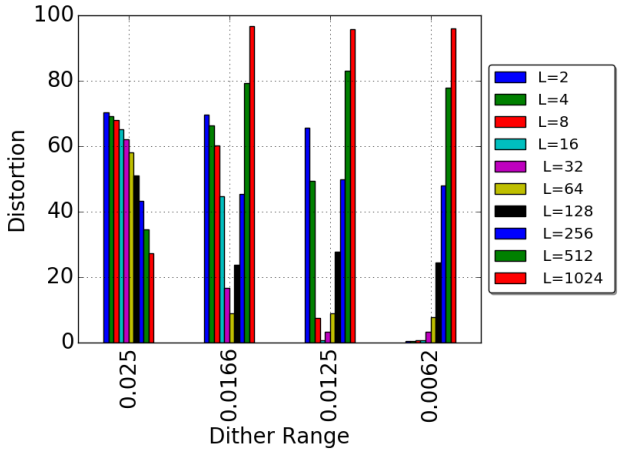


Figure 5.18: Localization distortion at $\Delta = 5cm, \sigma = 0.0$

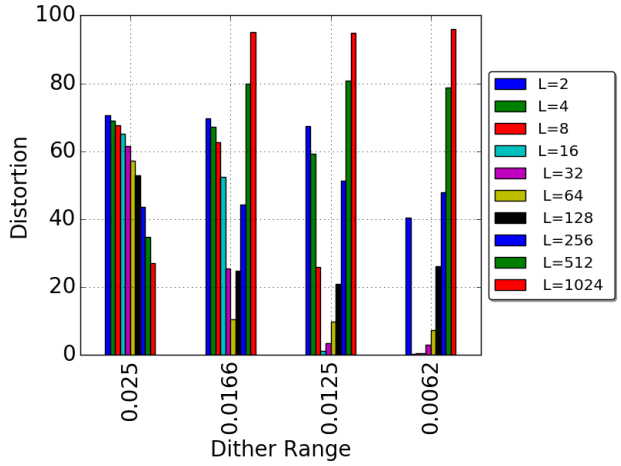


Figure 5.19: Localization distortion at $\Delta = 5cm, \sigma = 0.0072$

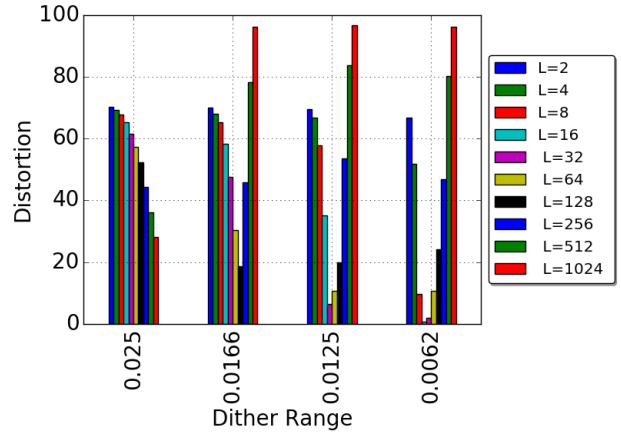


Figure 5.20: Localization distortion at $\Delta = 5cm, \sigma = 0.0144$

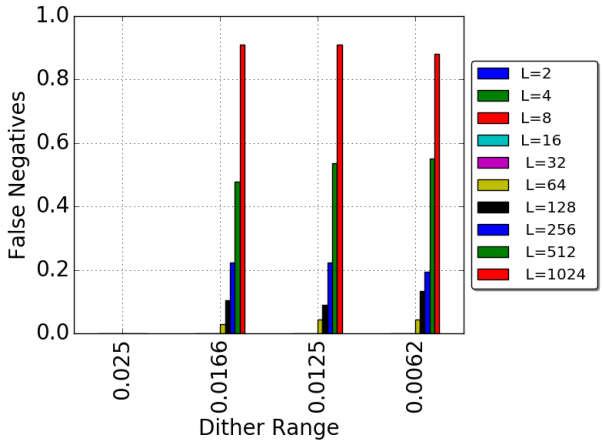


Figure 5.21: Detection False Negatives at $\Delta = 5cm, \sigma = 0.0$

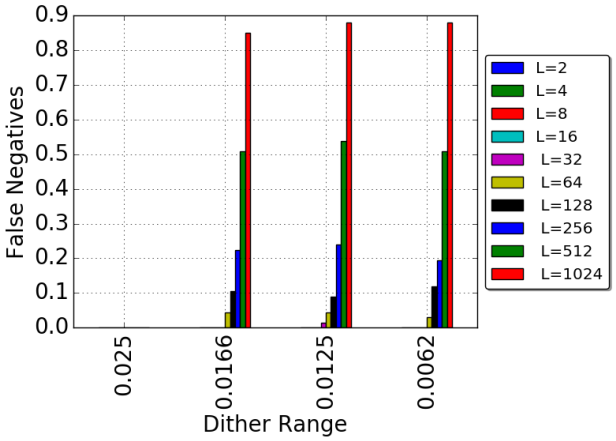


Figure 5.22: Detection False Negatives at $\Delta = 5cm, \sigma = 0.0072$

are false alarms, both positive and negative. In the case of sensor frame integrity checks, the false negatives cause more damage than false-positives. Any frame that is tampered and goes undetected by the system is not acceptable for our application. To get this value, both tampered and clean frames are given as input to the spread dither QIM pipeline.

To measure the false-alarms or the number of frames that are clean, but our algorithm flags them as tampered. The false-negative rate (f_n) is calculated as the ratio of number of tampered frames that are falsely determined as clean by the pipeline

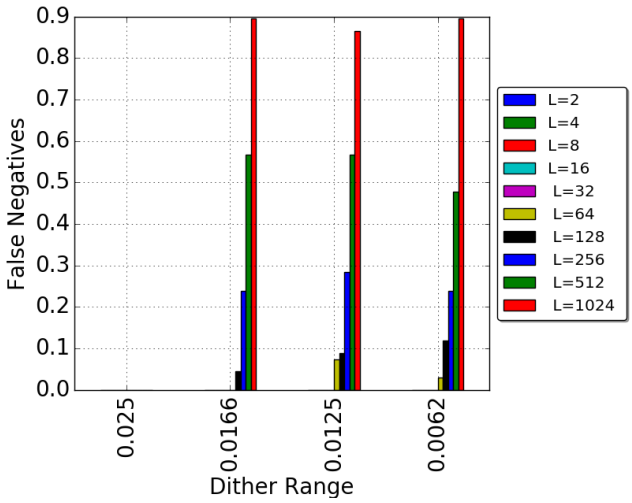


Figure 5.23: Detection False Negatives at $\Delta = 5cm, \sigma = 0.0144$

($N_{falseClean}$) to total number of tampered frames($N_{tampered}$) given as

$$f_n = N_{falseClean}/N_{tampered} \tag{5.10}$$

It is observed that the false-negative rate is close to 0% for dither range $\pm\Delta/2$, for lower dither ranges, it is low for lower spread lengths and drastically increases for higher spread lengths. Figure 5.21-5.23 show the average false negative values identified by the counter framework pipeline at different spread lengths and dither ranges.

CHAPTER VI

Future Work & Conclusion

In this chapter, we look into the future prospects of extending the research and provide concluding remarks on the work that has been done. When it comes to future work ideas, there are two different areas into which the research can be extended. One of them is to build a generic watermarking based data integrity framework to deal with different data transactions occurring in the automotive domain. The second idea is to extract the sensor-specific fingerprint and use it to build a security framework that can stop regular channel and the transmission channel attacks on the sensor data.

6.1 Need for Data Security in Autonomous Vehicles

Sensor data importance in modern cars and its integrity verification methods using cryptography alternatives like watermarking is discussed extensively in earlier chapters. Data transactions are happening all over the place in a modern car. Advances in driver assistance, automated driving, infotainment, and comfort features such as customization based on biometric sensors, etc are accounting for data generation and data transfers (Stanely, 2015). Along with individual vehicle generated data, the increase in connected vehicle technology such as 5G C2X and V2I/V2X where vehicles form clusters and share the data with vehicles inside the clusters is a growing trend.

According to a Gartner report, by 2022 the majority of premium segment cars would be offering connected vehicle services. With an average sensor set count of around 200 sensors measuring different data from the cars such as vehicle usage, vehicle environment, and driver biometrics, etc. a modern car acts like a high-end edge system with constant connectivity to the cloud and other vehicles in the vicinity (TE Connectivity, 2018). Along with the onboard sensing, these real-time communications with other vehicles and infrastructure help the cars to build temporal databases that help them with developing predictive route planning. All these on-board, off-board data transactions, flows need to be secured not only to provide safe and secured functionality but also to protect the data generator's intellectual property and protect them from false claims of liability. The cryptography based techniques can be of help to protect the integrity of the data to an extent but the end to end traceability of the data can only be achieved when we combine cryptography with watermarking.

To train the deep learning models that are used to make some automated driving decisions, data is collected from the real road driving of the vehicles. Companies spend huge amount of money to set-up vehicles, do test drives and to collect the drive data. Heavy load processing servers are used to store and retrieve the scenario information from these road runs. This data is accessed by engineers to train the models locally or on the cloud, some times the data is used to analyze the scenario better, to annotate it so the deep learning algorithm can be trained. The data transactions from the vehicle to cloud and cloud to local machines are secured but once the data is downloaded to a local node, it loses the security shield. With the current trend of remote working, there is no secure way to make sure that the data doesn't end up with some competitor who could use it to his advantage or train his models without having to spend money in collecting it. Watermarking can help answer most of the traceability, leakage control, access control questions. One of the growing research areas of watermarking is to bring it to the cutting edge technologies and as a

part of this research applying watermarking to real-time communications also needs to be researched. The time it takes to embed and extract a watermark along with its optimum performance is crucial when bringing this technology to real-time data transfers. In the above scenario embedding the vehicle data with a watermark before it gets uploaded to the cloud would provide end-to-end traceability but given the scenario where each vehicle is generating GigaBytes of data every hour, the time budget of watermark embedding, packetizing the data, and transporting becomes crucial. A robust watermarking scheme that can embed large volumes of data in such a short time is still a matter of research.

6.1.1 Data Sources

A McKinsey report states that current day cars have the compute power of 20 modern PCs, process about 25 GB of data/hr and contains around 100 million lines of code. Autonomous vehicles in particular are considered as supercomputers rolling on the highways, generating around 5 TB of data per hour, necessitating the need to split the data processing between the cloud and the edge which is the vehicle. Doing simple math, these mind-boggling data numbers can be easily proved. At a standard 30 fps data rate, a single video camera generates about 300 GB of data an hour for 720-pixel video and this number can creep up-to 5.4 TB for a 4K video resolution (Miller, 2020). Now consider an autonomous car, there are multiple sets of such cameras, along with other high data generators like a LiDAR and RADARS, etc. Many algorithms process this data in real-time and make self-driving decisions and some of the data is sent to the cloud for storage and post-processing and deep learning algorithm scenario training. Along with the sensor data the vehicles need to store and retrieve HD map data to help them identify the road boundaries, guard rails medians at centimeter-level accuracy. Along with the sensor data that senses the environment, a vehicle also carries many internal sensors to monitor its performance, this information needs to

be sent to the cloud as-well to support few specific applications. Many applications also require sensors to monitor human drivers, their cognitive distractions, and their ability to take over the vehicle in scenarios where the vehicle cannot drive by itself. These sensors collect data that need to be sent to the cloud for logging. Other customer convenience features also require the vehicle to collect driver information to customize the vehicle to his needs like setting the vehicle environment to driver needs.

6.1.2 Framework Proposal

We see that there is a need to bring in end-to-end transaction traceability, access control into data transactions in the automotive domain and this can be done with the help of watermarking techniques. We want to propose a generic framework to watermark any data contained in an automotive domain. If we consider the data that is collected by the sensors for the autonomous driving use case, it is estimated that around 30% of the data collected by the vehicle is will be uploaded to the cloud by every car. This accounts for the rest of 70% of the data to be processed in real-time by the vehicle or the edge node. The data that gets uploaded goes through lossy compression. Handing this data needs changes both in the cloud infrastructure as well as edge computing. Many interesting research problems such as how to manage the data storage and retrieval with low latencies, content and scenario mining, exchange of the data securely with different stakeholders, etc. arise as the companies start building the drive database. Here we present the high-level overview and try to provide a direction to future research in the area of data integrity and traceability.

To solve this problem, we first need to identify at what stage of this data flow process we can embed the watermark and what is the best strategy to embed the watermark.

If we take the general case of video data encryption, the mechanism chosen to

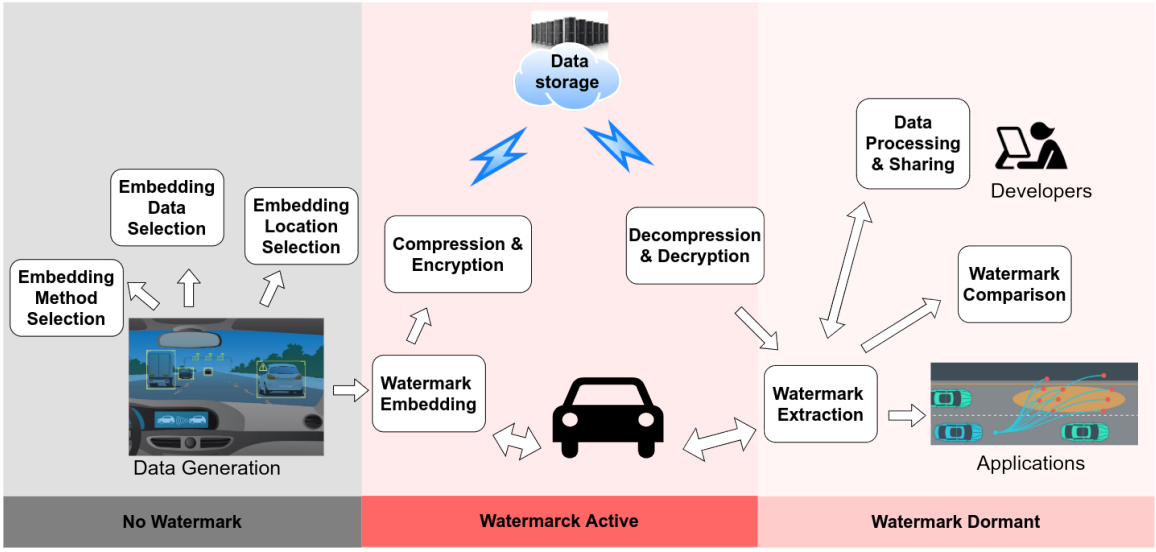


Figure 6.1: Data-transactions in modern-vehicles and watermarking framework proposal

encrypt the data where it is block cipher based or stream-based, etc. depends on the underlying universal coding standard that the video has to adhere to. In our use case, we do not have any format compliance or transcodability restrictions as the data compression strategy can be unique to a specific organization. Based on the data density, timing, and bandwidth requirements, they can make some design choices like partial encryption, encryption before or after compression, etc. but encryption and compression can be considered as inevitable steps in the data that gets transmitted to the cloud. We view adding watermarking to this equation would complement the cryptography and provides extended protection to the content even after the decryption. Now the question comes when should the data be watermarked.

Here there can be three choices to embed a watermark, *pre-encryption, and compression*, based on the robustness of the watermark, the watermarking could get altered or removed during the compression step. *Inter-encryption and compression*, here we find an appropriate step like quantization of the bitstream to embed the watermark during the compression and finally the *post-encryption and compression*,

again here application based careful consideration is required (Bohó et al., 2013). From these three choices, based on the computational power distribution and the time complexity we suggest going with the pre-compression and encryption stage for the watermark encryption. This method has a couple of advantages.

1. The data that doesn't get uploaded to the cloud can also benefit from the integrity checks as mentioned in this research

2. The watermark embedding burden is distributed among different sensors or the data origin locations so it's faster to encode and the concept of real-time watermarking can be achieved

The watermarking scheme we propose is the one that has high imperceptibility and can be universally applied to any type of data. QIM and its variants are primary candidates to investigate but this area can be researched to find any other spatial domain variants of the watermarking that fit the bill.

The next crucial step is to find the embedding location selection. By location, we mean wherein the data generation process we want to embed the watermark. Here we want to explore a universally applicable area. The embedding location should be data-independent, in the sense it should apply to all forms of data whether it is a raw video captured by high definition cameras or images or the 3D point cloud of the environment captured by a LIDAR or even a 1D time series captured by a biometric sensor, we should be able to apply the watermark universally. Here, without loss of generality, we can assume that any data generated from the vehicle is a set of packetized data. This assumption gives us the ability to apply the same watermarking technique at every vehicle data origin. If we look at data as the application-specific streams like frames for video, 3D point cloud for LiDAR data, etc, the embedding strategy must be specific to the sensor and soon the process can get clumsy with the addition of different sensors. Also, another assumption we make is that the data

generation nodes are connected to the vehicle network over Ethernet. Needless to say that Ethernet is becoming the major automotive network backbone. Most of the sensor manufacturers are also migrating to this network to support the bandwidth requirements. With this assumption, the proposal is to embed a watermark in different layers of the networking model as shown in Figure 6.2. Based on how we design the gateway modules and hopping networks within the vehicle, the embedded watermark can be stripped and re-assigned or it can be directly forwarded to the destination node depending on the layer in which the watermark is embedded. Also, to let the watermark stay beyond the network transactions, the watermark needs to be randomly embedded into the user data itself as a data element. In (Artru et al., 2019) multiple locations to embed the watermark in-network headers are discussed, such as

- **Physical layer embedding**: At this level, the data bits are arranged as bit frames before getting transmitted as a signal on the Ethernet bus. This layer can be used to embed data by using spread spectrum techniques as discussed in (Li et al., 2013).

- **Storage channel embedding**: The storage channel watermarking exploits the redundancies and the unused fields of the multi-layer Ethernet stack shown in Figure 6.2. In (Kundur and Ahsan., 2003), a method to store a bit per datagram by modifying the 3-bit flag bit in the IP header. Similarly, making use of the Time to Live, TCP sequence numbers, packet length alterations, and checksum packets are discussed in (Collins and Agaian, 2016).

- **Timing channels**: Using the time sync modules to hide and forward the watermark between two endpoints is discussed in (Houmansadr et al., 2009). One such simple method is modifying the inter-packet Delay (IPD) to embed the watermark. There are also methods to exploit the TCP segment temporal bursts

(Luo et al., 2008).

- **Application protocols**: Many applications are built on the top of the network stack such as FTP, SMTP, HTTP, SSH, etc. The choice of which is dependent on the end application. These applications can have an optional header file that can be added to the user data and this can be used to embed the watermark. Several methods like using secret fields to inform receivers that the MAC contains the watermark as discussed in (Lucena et al., 2005). The works that mention ways to embed the watermark in the spaces included in HTTP headers reveal that there exist multiple locations to store a cleverly crafted watermark in any application.

- **User data**: Given the packet size of an Ethernet frame, many watermark embedding schemes like the addition of watermark to the packet data at random locations can be explored. This method has the advantage that the watermark stays with the data beyond the data stripping realm of the network.

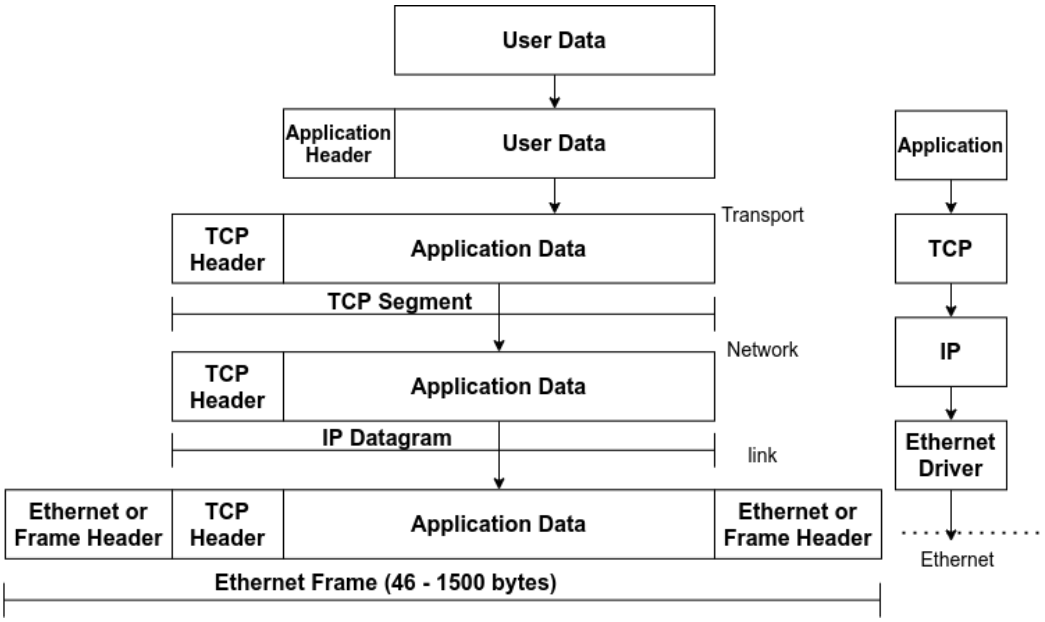


Figure 6.2: OSI model of an Ethernet frame

The proposed framework can make use of spatial domain watermarking methods like

the QIM and can be worked on as an extension to the concepts verified in this work. As shown in Figure 6.1, the framework can be divided into different areas like watermark embedding method selection where you decide the best approach to insert watermark into the data, then decide the watermark location like where int e network layer you want to insert the watermark and what type of data you want to watermark. Here the autonomous car collects data from on-board sensors, external entities like other vehicles, map servers, etc. This data gets watermarked at the source and then can be consumed within the vehicle or the edge for real-time data interpretation or can be uploaded to the cloud after compression. After the extraction process, the data can be used for post-processing like deep learning model training or sharing with multiple stakeholders or within the vehicle where the watermark will be still in the dormant state ready to be retrieved if required.

6.2 Sensor Fingerprints

As the market for autonomous vehicles advances, a need for robust safety protocols also increases. Autonomous vehicles rely on sensors to understand their operating environment. Active sensors such as cameras, LiDAR, ultrasonic, and radar are vulnerable to regular channel attacks. One way to counter these attacks is to pattern match the sensor data with its unique physical distortions, commonly referred to as a fingerprint. This fingerprint exists because of how the sensor was manufactured, and it can be used to determine the transmitting sensor from the received waveform.

Fingerprints are formed due to microscopic imperfections and dissimilarities in the sensor manufacturing process. They are physical features prevalent in a multitude of Cyber-Physical Systems (CPS) and other hardware devices that arise in specific waveform characteristics. Sensor fingerprints can be represented as a function of the material properties which make up a sensor or a piece of hardware and fabrication process. These imperfections are assumed to be unique to a specific sensor and random.

The concept of physical fingerprinting has been used for RF transmitter identification (Deng et al., 2017) and hardware validation for sensors used in non-automotive applications (Ahmed et al., 2020). In this study, we focus on extracting sensor intrinsic properties called **fingerprints** that can serve as a potential countermeasure for two physical signal level attacks, which are attacks categorized by manipulating the environment in such a way to cause incorrect ultrasonic sensor measurements. Using an ultrasonic sensor, we establish that there exists a specific distortion profile in the transmitted waveform called physical fingerprint that can be attributed to their intrinsic characteristics. In the case of ultrasonic sensors, this fingerprint manifests in the form of random noise in the transmitted pulse sequence from the sensor which can be observed in the sensor transmissions.

We propose a joint time-frequency analysis-based framework for ultrasonic sensor fingerprint extraction and use it as a feature to train a Naive Bayes classifier. The trained model is used for transmitter identification from the received physical waveform. In the future, this proposed framework can be extended using a simple or deep learning-based classifier to identify its signature in the returns and reject data from an attacker.

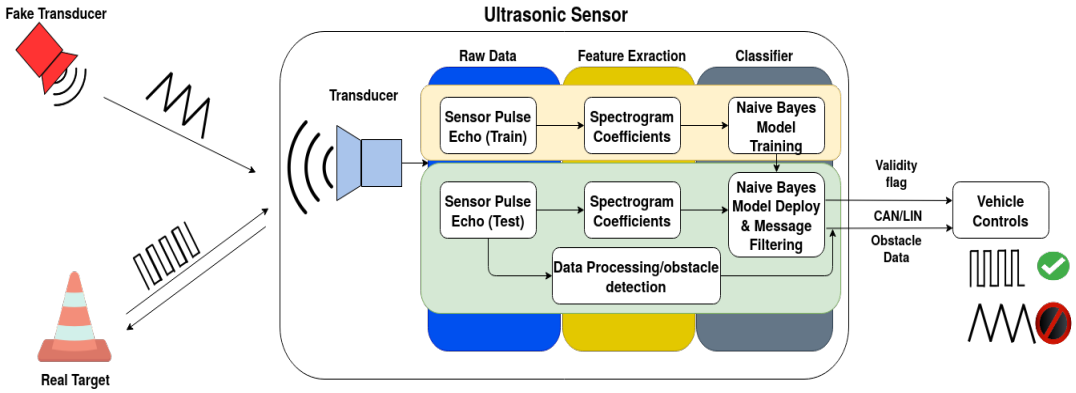


Figure 6.3: Block-diagram of the system model

6.2.1 Methodology

We propose a method to extract the sensor fingerprints by observing the spectrogram for each sensor at multiple distances to determine each sensor's respective resonance frequency Figure 6.5. Once we have determined the resonance frequency for a sensor, our algorithm extracts data from the necessary frequencies which will create a frequency profile used for training our classifier. By applying a band-pass filter to our data, our classifier ignores irrelevant data and is in turn more accurate. Each frequency bin in our spectrogram over the desired interval acts as a feature vector later for our classifier and essentially contains the fingerprinting information of a given sensor. As mentioned, pattern matching a fingerprint to a specific sensor based on spectral content is extremely robust, since it is infeasible for an attacker to generate and transmit an ultrasonic waveform with the same fingerprint or random noise profile, even if the attacker has a sophisticated knowledge of our implementation. Here, we train a simple, computationally light machine learning model with this feature to demonstrate that the transmitting sensor can be identified through a physical fingerprint.

6.2.2 System Model

The system model assumes an ultrasonic sensor system on chip devices commonly used in automotive applications (Texas Instruments, 2014). The sensor does the signal conditioning and processing for the transducer echo signals and transmits the distance to the obstacle and other parameters over the chosen interface like CAN, LIN. The on-board ECU allows complete configurability for the end applications.

The proposed fingerprint extraction happens on the sensor itself, during an initial calibration phase where the sensor learns the fingerprint and trains a model to identify its echo and differentiate it from others. This model can be used at a later stage to identify if the sensor is under attack. Shown in Figure 6.3 is the block diagram of the

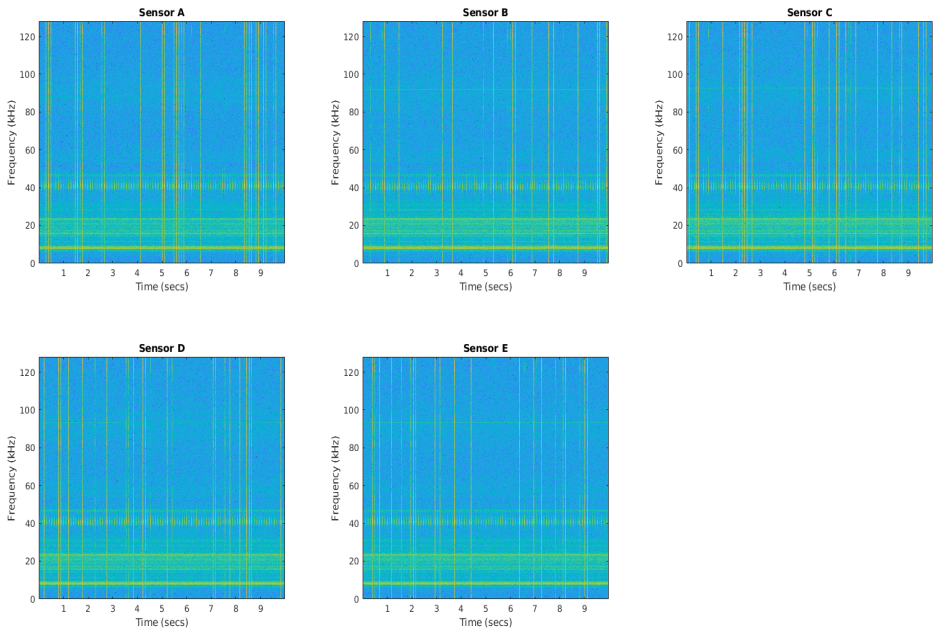


Figure 6.4: Spectrograms of sensor transmissions generated using 8 ms window size, 25% overlap, and Hanning weight window

system model. During the calibration phase, the system learns its echo and trains the model which is then used to determine the authenticity of the received signal. Specifically, the received signal is analyzed for fingerprint extraction in the background while the data gets processed to detect obstacles. The output from the sensor includes a validity flag along with the data to assure that the data is authentic and not subject to physical attacks. In the proposed framework, we use power spectrum coefficients as features and a simple Gaussian Naive Bayes classifier to perform supervised learning and classification of labeled data. As the Naive Bayes classifier supports multi-class classification, it will not only allow our system to accurately detect when an attack occurs but also on what sensor, since most vehicles that utilize ultrasonic sensors use more than one.

Our system for combating attacks launched by the adversary is under the assumption that the time in which we detect an attack is not a leading factor in the success of our model. In real-time applications, ADAS systems have stringent safety require-

ments such as brake engagement that have a maximum latency of 0.1 seconds (Lin et al., 2018).

6.2.3 Data Model

Our data model assumes that the data inputs have the following characteristics as noted in (Xu et al., 2018) except we define the transmitted and received signals with the inclusion of noise characteristics emitted by the transducer due to a hardware fingerprint. We can describe the transmitted waveform of our ultrasonic sensor as an ideal sinusoidal signal

$$s(t) = A\cos(\omega_c t), \quad t\epsilon[0, \infty] \tag{6.1}$$

Where in Equation (6.1), A is the amplitude of the signal, t is the time and ω_c is the radial frequency of the carrier signal. In reality, the transmitted signal will have some noise component to it as a result of the hardware fingerprint

$$s_r(t) = A\cos(\omega_c t) + n_r(t), \quad t\epsilon[0, \infty] \tag{6.2}$$

Where in Equation (6.2), $n_r(t)$ denotes the noise of the transmitted signal due to the hardware fingerprint. At the receiver, the transmitted signal appears as

$$r(t) = \alpha\cos((\omega_c t + \omega_D)(t - \tau) + \theta) + n_r(t) + n(t), \quad t\epsilon[0, \infty] \tag{6.3}$$

Where in Equation (6.3), α represents the attenuated amplitude of the transmitted signal, ω_D is the Doppler velocity, τ is the time delay (time for the echoed signal to reach the receiver), θ is the phase shift, and $n(t)$ is the additive noise component. We expect $n_r(t)$ to be centered at the resonance frequency of our sensors since ultrasonic sensors transmit pulses by exciting a piezoelectric transducer (Hagood and von Flotow, 1991). This transducer will vibrate acoustically at the same frequency as the AC

voltage that is applied to it. Noise due to microscopic hardware imperfections will be exacerbated around the resonance frequency of the sensor. Signals of this type will be analyzed and used as input to our classifier.

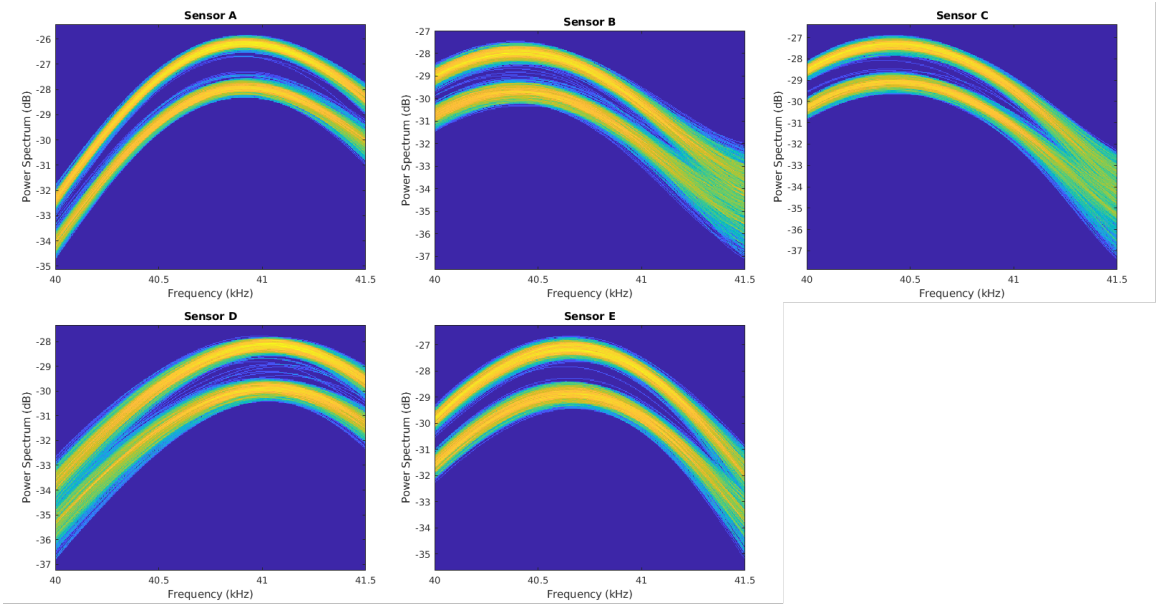


Figure 6.5: Power spectrum of sensors under test at 25 cm distance measurement

6.2.4 Threat Model

While evaluating the security of our model, it is important first to identify what possible adversaries we must defend against and what types of attacks they can employ. We identified these main physical channel attacks on the ultrasonic sensors. It is assumed that the attacker will be able to perform these three types of attacks and for launching these attacks, the assumption is made that the attacker will have a know-how of all the information of our system, such as what sensors are used, the frequency at which data is recorded, and even our method for defending against malicious attacks.

1. *Jamming Attacks*: The attacker will be able to perform jamming attacks (Li et al., 2007), where the transducer of an ultrasonic sensor is always excited with

ultrasound in such a way that it cannot measure the echo of its transmitted ultrasonic waves and therefore cannot accurately perceive its surroundings.

2. *Spoofing Attacks*: The attacker will be able to generate ultrasonic pulses to excite the transducer of an ultrasonic sensor such that a "phantom object" can be perceived by the sensor when it is not truly there. This is the case when an ultrasonic wave is spoofed to the transducer of an ultrasonic sensor before the echo of its transmitted wave can return, resulting in the sensor perceiving a non-existent object. Although this is difficult to perform while a sensor is in motion due to timing dependencies, it has been implemented on stationary sensors used in automobiles in (Chen. Yan, 2016) in the case where the attacker has knowledge of the frequency of ultrasonic sensor readings, which fits this threat model.

3. *Sensor Damage & Replacement*: In addition to jamming and spoofing attacks, the adversary may also perform an attack that requires physical contact with the sensor. This is the case when the adversary damages (Elvin et al., 2003) the sensor or replaces it entirely. It is assumed that the adversary can do this stealthily, such that visually it is not possible to tell whether or not a sensor has been physically damaged, replaced, or altered in any way.

The proposed framework can handle jamming and spoofing attacks along with the sensor damage contact-based attacks. Since we assume a smart sensor that runs the data-processing on-board, we cannot detect the sensor replacement contact-based attack.

6.2.5 Fingerprint Extraction

To extract and localize the hardware-specific fingerprints, we chose the time-frequency analysis method. As spectrograms give the time-frequency distribution

of time series data, we started with spectrogram analysis of the sensors under test. In Figure 6.4, the spectrograms of the five sensors under test are shown. With the reduced window size of 8 ms, the frequency distribution of each sensor is visually distinguishable, although minutely, and laid the first step towards our claim towards the presence of an intrinsic sensor fingerprint. As a next step, we focused on the spectral components at a central frequency of operation of the sensors. We obtained the power spectrum of the ultrasonic sensor signal around the operating frequency of 40 kHz with a timing resolution of 250 ms and a frequency resolution of 1 kHz. The power spectrum is generated with a persistence option to visualize the percentage of time that a particular frequency component is present in the input signal. The results as shown in Figure 6.5, display a distinct feature in the form of the power spectrum peak location that can be used to identify each sensor. The power spectrum peak and the corresponding peak shape profile occurred at different frequencies for different sensors under test. It can be observed from the Figure 6.5, that the spectral peaks for sensors under test, A,B,C,D & E occurred at 40.91, 40.36, 40.45, 41.03, 40.65 kHz respectively. The peak locations of any two sensors were separated with a 100 Hz frequency resolution and the peak roll-off rates for different sensors are different as-well. Given the fact that our sensors under test are from the same manufacturer, of the same grade and data collection conditions are the same across multiple experiment runs, the variation in the location of the peak for power spectral components can be considered as a unique fingerprint for each sensor. We used this variation in the peak location and the shape profile information as our main feature for the classification of the sensors. Though it can be argued that as the number of sensors increases drastically the frequency resolution might not be sufficient to distinguish different sensors based on just the spectral peak location, for our end application of supporting Advanced Driver Assistance System (ADAS) or Automated Driving (AD) features, the number of ultrasonic sensors used in a vehicle is usually less than 15. For

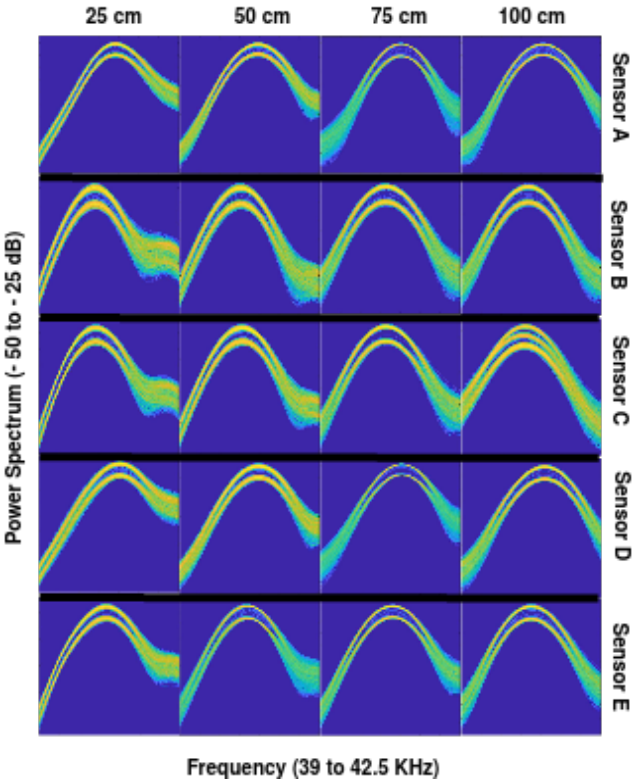


Figure 6.6: Power spectrum of all sensors under test at different distances

instance, Tesla autopilot advanced sensor coverage has 12 ultrasonic sensors (Tesla, 2020). We observed similar trends in power spectrum peak location and shape profile at different distances as shown in Figure 6.6. The power spectrum visualization in Figure 6.6 shown in a table form with each row displaying the power spectra of a single sensor collected at different distances and similarly the columns represent the power spectra of different sensors at a given distance. While the peak location was a good feature to classify different sensors at a given distance it did not generate good results for distance agnostic sensor classification. It can be observed that for distance agnostic sensor classification feature the peak roll-off rate and the shape profiles need to be used and modeled. This is considered a future extension of this research.

6.2.6 Experiments & Results

The first step in building a system model to counter the physical attacks on an ultrasonic sensor is to establish that different ultrasonic sensors generate fingerprints in their transmissions unique to the host and this fingerprint can be used to identify the host sensor. To prove this point, we set-up an experiment as shown in Figure 6.7. The microphone placed at various distances from an ultrasonic sensor captures the sensor transmissions and records them.

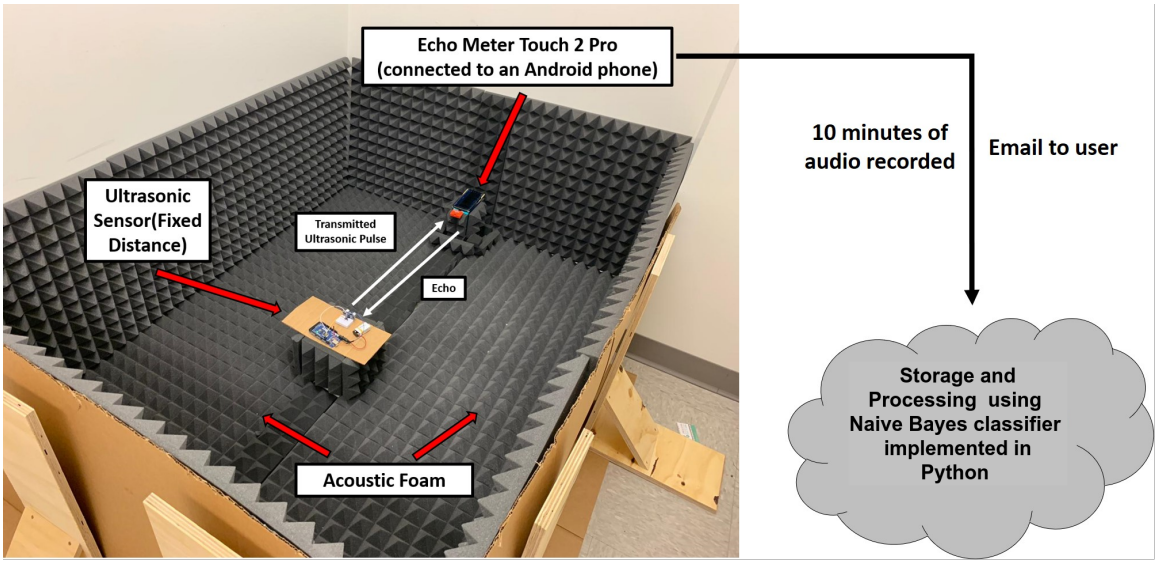


Figure 6.7: Data collection set-up for fingerprint extraction

This recorded data is then used to generate feature vectors from spectrograms. After generating the scattering features for each ultrasonic sensor under test, the Gaussian Naive Bayes model is trained with the training dataset. The Gaussian Naive Bayes classifier also had promising results as shown in Table 6.1. Data used from the same experiment is shown in Figure 6.7. was input to our classifier. One benefit of the Gaussian NB method is that only 10 percent of the data was needed for training to achieve high accuracy classification.

As an extension, we decided to synthetically saturate the received signal of our ultrasonic sensor by adding a percentage of the peak noise values seen graphically

Distance (cm)	Training Size	Test Size	Accuracy
25	10%	90%	99.67%
50	10%	90%	96.68%
75	10%	90%	95.42%
100	10%	90%	99.66%
Mixed Distances	10%	90%	91.72%

Table 6.1: Accuracy: Gaussian NaiveBayes Method

in Figure 6.4 as a DC component to the signal before the spectrogram is applied. By synthetically adding Gaussian white noise (Simon, 2002), the discernibly of the fingerprint was diminished. The goal of this was to experiment with pseudo-jamming to see at what point our classifier would no longer be able to successfully identify a sensor. To recursively add noise until the fingerprint was no longer identifiable, we let the amount of saturation be proportional to some value of the peak value.

$$N_r[n] = x[n] + \alpha\sigma_x N[n] \tag{6.4}$$

Where $x(n)$ is the received digital signal,σ_x is the standard deviation of the original signal, α is a saturation coefficient and $N[n]$ is a noise signal with standard normal mean and standard deviation and $N_r[n]$ is our total saturation which is added to the entire time-domain signal. Figure 6.8 shows the effect different values of α have on the spectrum of the received signal. The classifier performed reasonably well for

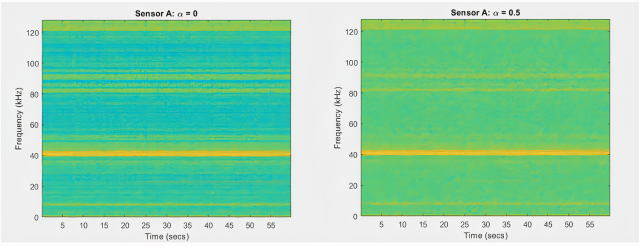


Figure 6.8: Saturation of received signal - Spectrogram visual

values of $0 \leq \alpha \leq 0.539$. The Table 6.2 below shows the accuracy of the classifier for different values of α for a single sensor

Distance (cm)	α_{max}	Accuracy
25	0.539	91.41%
50	0.4179	93.38%
75	0.4009	92.59%
100	0.394	90.33%
Mixed Distances	0.2154	91.72%

Table 6.2: Accuracy: Saturated Gaussian NaiveBayes Method

This framework of fingerprint extraction can be combined with watermarking to build an end-to-end sensor security system that can work against regular channel and transmission channel attacks.

6.3 Conclusion

In this research, we presented the watermarking methods to deal with the insider attacks on the sensor data that pose a real threat to autonomous vehicles. A Cyber-physical system, such as an autonomous vehicle, is susceptible to insider attacks targeting sensors and their transmission channels, making it necessary to verify the integrity of sensor data before acting on it. Traditional data integrity protection methods like cryptography cannot be applied in their entirety to solve this problem due to their resource requirements and complexity. In chapters 2 & 3 we presented an overview of the watermarking methods. In chapter 4, we proposed a 2D QIM for watermarking the RADAR data and proposed a pipe-line to design watermarks when dealing with smart sensors. This pipeline is tested for the effects of embedding induced distortion using simulated RADAR data on an EKF based sensor fusion algorithm. The experimental results conclude that that the 2D QIM method for watermarking has a little or no effect on the EKF predictions for small values of quantization step-size $\Delta \leq 0.05\ m$, which can be attributed to the minimal distortion induced by the 2D QIM process. In chapter 5, we proposed a novel approach to detect and localize tampering of the raw data from the LiDAR sensor. We demonstrated that

the proposed method can detect and localize tampering to the real-world benchmark KITTI dataset with a 100% success rate as long as additive noise is less than the quantization step-size. We also established the QIM embedding-induced distortion thresholds for proper detection using 3D FCN and VoxelNet deep learning models. We analyzed the security vulnerabilities of the plain QIM method and enhanced it by proposing a spread dither 3D QIM method to verify the sensor data integrity. We tested the effectiveness of the proposed method on KITTI LiDAR sensor data-set. We deduced the optimum values of parameters for the proposed method by building a pipe-line based on the proposed approach to detect and localize the tampering of the raw data from the sensor.

The low complexity data hiding or watermarking techniques proposed in this research can be applied to existing legacy interfaces without burdening the interface bandwidth or computational resources of the system. This makes the process of transitioning to the secured data link possible even in legacy systems. Always, having some level of security in place is still better than having none. Having sensor data integrity checks in place can help to secure the applications and build safer systems. In the security world, often a layered architecture is preferred wherein if an attack cannot be prevented; It can be detected to prevent the worst outcome. We believe that watermarking the sensor data adds another layer to the security scheme using some light-weight and yet efficient techniques. These techniques can be used either in a standalone mode or in conjunction with traditional cryptography methods wherever necessary, to secure data transfers over any physical interface such as CAN/CAN-FD, Ethernet, etc.

We also presented the future directions to the research. This research can be extended to designing a universal framework to provide integrity verification and traceability to different data transactions in the automotive domain. Also, we propose to exploit the unique watermarks for sensors by extracting the sensor intrinsic

distortions or fingerprints that can be successfully used to identify them. The proposed frameworks can be extended to different sensor modalities, different watermark embedding methods along with the study to find out the effects of embedding induced distortion on more complex process models and state vectors.

APPENDICES

APPENDIX A

Algorithms

Algorithm 1: Find Modified Indices

Result: $modifiedIndices[\,]$
$modifiedIndices \leftarrow 0$;
$gtlistIndex \leftarrow 0$;
while $gtIndex < len(gtlist)$ **do**
    if $gtlist[gtlistIndex] == modlist[modlistIndex]$ **then**
        $gtlistIndex+ = 1$;
    end
    $modifiedIndices[\,] \leftarrow gtlistIndex$;
    $gtlistIndex+ = 1$;
end

Algorithm 2: Find Deleted Indices

Result: $missingIndices[\,]$
$missingIndices \leftarrow 0$;
$gtlistIndex \leftarrow 0$;
$modlistIndex \leftarrow 0$;
while $modlistIndex < len(modlist)$ *&* $gtIndex < len(gtlist)$ **do**
    if $gtlist[gtlistIndex] == modlist[modlistIndex]$ **then**
        $gtlistIndex += 1$;
        $modlistIndex += 1$;
    end
    $missingIndices[\,] \leftarrow gtlistIndex$;
    $gtlistIndex += 1$;
end

Algorithm 3: Find Added Indices

Result: $addedIndices[\,]$
$addedIndices \leftarrow 0$;
$gtlistIndex \leftarrow 0$;
$modlistIndex \leftarrow 0$;
while $modlistIndex < len(modlist)$ *&* $gtIndex < len(gtlist)$ **do**
    if $gtlist[gtlistIndex] == modlist[modlistIndex]$ **then**
        $gtlistIndex += 1$;
        $modlistIndex += 1$;
    end
    $addedIndices[\,] \leftarrow gtlistIndex$;
    $modlistIndex += 1$;
end

Algorithm 4: Watermark Sequence Generator

Result: $generatedSequence[\,]$
$generatedPair \leftarrow LSbits(b(t_0))$;
$generatedSequence[\,] \leftarrow generatedPair$;
$randomNum \leftarrow pseudorandom(MSnibble(b(t_0)))$;
$num_{Pairs} \leftarrow floor(randomNum)$;
while $size(generatedSequence) <= num_{Pairs}$ **do**
    $generatedPair += b01$;
    $generatedPair = generatedPair\%4$;
    $generatedSequence[] \leftarrow generatedPair$;
end

APPENDIX B

Source Code

All the git repositories used for the development of source code for this research are made public.

1. 2D QIM implementation for RADAR sensor from Chapter 4, can be found in the following github location (Changalvala et al., 2020)
   https://github.com/raghu429/RADAR_DataIntegrity.git

2. 3D QIM implementation for LiDAR sensor from Chapter 5, can be found in the following github location (Changalvala and Malik, 2019b), (Changalvala and Malik, 2019a)
   https://github.com/raghu429/LiDAR_QIM.git

3. 3D Dither QIM implementation for LiDAR sensor from Chapter 5, can be found in the following github location (Changalvala and Malik, 2020)
   https://github.com/raghu429/DitherQIM.git

4. Fingerprint generation for Ultrasonic sensor from Chapter 6, can be found in the following github location (Cheek et al., 2020)
   https://github.com/raghu429/Ultrasonic_Integrity.git

BIBLIOGRAPHY

Agarwal, P., and B. Prabhakaran (2009), "Robust blind watermarking of point-sampled geometry," *IEEE Trans. on Information Forensics and Security*, vol. 4(1), pp. 36-48.

Ahmed, C. M., A. P. Mathur, and M. Ochoa (2020), "Noisense: Detecting data integrity attacks on sensor measurements using hardware-based fingerprints," *ACM Transactions on Privacy and Security*, 24(1), doi: 10.1145/3410447.

Artru, R., A. Gouaillard, and T. Ebrahimi (2019), "Digital watermarking of video streams: Review of the state-of-the-art," *ArXiv*, vol. abs/1908.02039.

AUTOSAR CP R19-11 (2019), "Specification of time synchronization over CAN, standard," *AUTOSAR*, AUTOSAR CP R19-11.

AUTOSAR CP Release 4.3.1 (2017), "Specification of secure onboard communication, standard," *AUTOSAR*, AUTOSAR CP Release 4.3.1.

Baxter, J. A., D. A. Merced, D. J. Costinett, L. M. Tolbert, and B. Ozpineci (2018), "Review of electrical architectures and power requirements for automated vehicles," *IEEE Transportation Electrification Conference and Expo*, pp. 944-949.

Begum, M., and M. S. Uddin (2020), "Digital image watermarking techniques: A review," *Information*, 11(2), doi: 10.3390/info11020110.

Bitar, A., R. Darazi, J.-F. Couchot, and R. Couturier (2017), "Blind digital watermarking in pdf documents using spread transform dither modulation," *Multimedia Tools and Applications*, 76, pp. 143-161, doi: 10.1007/s11042-015-3034-2.

Boho, A., G. Wallendael, A. Dooms, J. De Cock, G. Braeckman, P. Schelkens, B. Preneel, and R. Van de Walle (2013), "End-to-end security for video distribution: The combination of encryption, watermarking, and video adaptation," *IEEE Signal Processing Magazine*, vol. 30, pp. 97-107, doi: 10.1109/MSP.2012.2230220.

Brian, C., and W. G. W (2001), "Quantization index modulation methods for digital watermarking and information embedding of multimedia," *Journal of VLSI Signal Processing Systems for Signal, Image, and Video Technology*, 27(1-2), pp. 7-33, doi: 10.1023/A:1008107127819.

Cárdenas, A. A., S. Amin, Z.-S. Lin, Y.-L. Huang, C.-Y. Huang, and S. Sastry (2011), "Attacks against process control systems: Risk assessment, detection, and response," *6th*

ACM Symposium on Information, Computer and Communications Security, pp. 355-366, doi: 10.1145/1966913.1966959.

Changalvala, R., and H. Malik (2019a), "LiDAR data integrity verification for autonomous vehicle," *IEEE Access*, vol. 7, pp. 138,018-138,031, doi: 10.1109/ACCESS.2019.2943207.

Changalvala, R., and H. Malik (2019b), "LiDAR data integrity verification for autonomous vehicle using 3d data hiding," *IEEE Symposium Series on Computational Intelligence*, pp. 1219-1225, doi: 10.1109/SSCI44817.2019.9002737.

Changalvala, R., and H. Malik (2020), "Sensor data integrity verification for autonomous vehicles using spread 3D dither QIM," *IEEE 92nd Vehicular Technology Conference*, pp. 1-7, doi: 10.1109/VTC2020-Fall49728.2020.9348492.

Changalvala, R., B. Fedoruk, and H. Malik (2020), "Radar data integrity verification using 2D QIM-based data hiding," *Sensors*, 20(19), 5530, doi: 10.3390/s20195530.

Checkoway, S., et al. (2011), "Comprehensive experimental analyses of automotive attack surfaces," *USENIX Conference on Security*.

Cheek, E., D. Khuttan, R. Changalvala, and H. Malik (2020), "Physical fingerprinting of ultrasonic sensors and applications to sensor security," *IEEE 6th International Conference on Dependability in Sensor, Cloud and Big Data Systems and Application*, pp. 65-72, doi: 10.1109/DependSys51298.2020.00018.

Chen, B., and G. W. Wornell (1998), "Digital watermarking and information embedding using dither modulation," *IEEE 2nd Workshop on Multimedia Signal Processing*, vol. 1998-Decem, pp. 273-278, doi: 10.1109/MMSP.1998.738946.

Chen, B., and G. W. Wornell (2001), "Quantization index modulation: A class of provably good methods for digital watermarking and information embedding," *IEEE Transactions on Information Theory*, 47(4), pp. 1423-1443, doi: 10.1109/18.923725.

Chen, X., H. Ma, J. Wan, B. Li, and T. Xia (2017), "Multi-view 3D object detection network for autonomous driving," *30th IEEE Conference on Computer Vision and Pattern Recognition*, vol. 2017-Janua, pp. 6526–6534, doi: 10.1109/CVPR.2017.691.

Chen, Y., J. Liu, W. Xu (2016), "Can you trust autonomous vehicles : Contactless attacks against sensors of self-driving vehicle," *DEF CON*, vol. 24, pp. 1-13.

Chopra, J., A. Kumar, A. K. Aggarwal, and A. Marwaha (2018), "An efficient watermarking for protecting signature biometric template," *5th International Conference on Signal Processing and Integrated Networks*, pp. 413-418, doi: 10.1109/SPIN.2018.8474269.

Cleland, A. N. (2003), "Two-and three dimensional lattices," in *Foundations of Nanomechanics,* Berlin: Springer-Verlag, pp. 43-85 doi: 10.1007/978-3-662-05287-7.

Collins, J., and S. Agaian (2016), "Trends toward real-time network data steganography," *ArXiv*, vol. abs/1604.02778.

Conway, J., and N. Sloane (1982), “Fast quantizing and decoding algorithm for lattice quantizers and codes,” *IEEE Transactions on Information Theory*, vol. 28, pp. 227-232, doi: 10.1109/TIT.1982.1056484.

Cox, I., M. Miller, J. Bloom, J. Fridrich, and T. Kalker (2008), *Digital Watermarking and Steganography*, 2 ed., San Francisco: Morgan Kaufmann Publishers Inc.

Cui, J., L. Liew, G. Sabaliauskaite, and F. Zhou (2018), “A review on safety failures, security attacks, and available countermeasures for autonomous vehicles,” *Ad Hoc Networks,* vol. 90, doi: 10.1016/j.adhoc.2018.12.006.

Deng, S., Z. Huang, X. Wang, and G. Huang (2017), “Radio frequency fingerprint extraction based on multidimension permutation entropy,” *International Journal of Antennas and Propagation*, pp. 1-6, doi: 10.1155/2017/1538728.

Digital Watermarking Alliance (2020), “Digital watermarking applications,” [Online]. Available: https://digitalwatermarkingalliance.org/digital-watermarking-applications, [Accessed: Dec. 12, 2020].

Dzemyda, G., O. Kurasova, J. Zilinskas (2013), “Multidimensional data and the concept of visualization,” in *Multidimensional Data Visualization: Methods and Applications,* NewYork:Springer, pp. 1-4, doi: 10.1007/978-1-4419-0236-8_1.

European Commission (2017), “Autonomous cars: A big opportunity for european industry,” *Digital Transformation Monitor- European Commission*, [Online]. Available:https://ati.ec.europa.eu/reports/sectoral-watch/autonomous-cars-big-opportunity-european-industry.

Elvin, N., et al. (2003), “A self-powered damage detection sensor,” *The Journal of Strain Analysis for Engineering Design*, vol. 38(2), doi: 10.1243/030932403321163640.

Feng, J., and M. Potkonjak (2003), “Real-time watermarking techniques for sensor networks,” *SPIE 5020 -Security and Watermarking of Multimedia Contents V*, doi: 10.1117/12.479736.

Gope, P., and T. Hwang (2016), “BSN-care: A secure IoT-based modern healthcare system using body sensor network,” *IEEE Sensors Journal*, vol. 16(5), pp. 1368-1376, doi: 10.1109/JSEN.2015.2502401.

Hagood, N. W., and A. von Flotow (1991), “Damping of structural vibrations with piezoelectric materials and passive electrical networks,” *Journal of Sound and Vibration*, vol.146, pp. 243-268, doi: 10.1016/0022-460X(91)90762-9.

Hanson, A. J. (1994), “Geometry for n-dimensional graphics,” in *Graphics Gems IV,* San Diego: Academic Press Professional, pp. 149-170, doi: 10.1016/B978-0-12-336156-1.50024-0.

Hartmann, K., and C. Steup (2013), “The vulnerability of uavs to cyber attacks - an approach to the risk assessment,” *in 5th International Conference on Cyber Conflict,* pp. 1-23.

Houmansadr, A., N. Kiyavash, and N. Borisov (2009), "Rainbow: A robust and invisible non-blind watermark for network flows," *Network and Distributed Security Symposium Symposium*.

Huang, Q. (2018), *Voxelnet*, GitHub repository, [Online]. Available: https://github.com/qianguih/voxelnet, commit-b74823daa328fc2fa99452bf79793e1f3c32c72a.

Jetto, L., S. Longhi, and G. Venturini (1999), "Development and experimental validation of an adaptive extended kalman filter for the localization of mobile robots," *IEEE Transactions on Robotics and Automation*, 15(2), pp. 219-229.

Jo, K., J. Kim, D. Kim, C. Jang, and M. Sunwoo (2014), "Development of autonomous car part i: Distributed system architecture and development process," *IEEE Transactions on Industrial Electronics*, vol. 61(12), pp. 7131-7140, doi: 10.1109/TIE.2014.2321342.

Jo, K., J. Kim, D. Kim, C. Jang, and M. Sunwoo (2015), "Development of autonomous car part ii: A case study on the implementation of an autonomous driving system based on distributed architecture," *IEEE Transactions on Industrial Electronics*, vol. 62(8), pp. 5119-5132, doi: 10.1109/TIE.2015.2410258.

Joachim, E., and G. Bernd (2002), "General concepts and state-of-the-art," in *Informed Watermarking*, Boston: Springer, pp. 7-31.

Kaur, G., S. Singh Kasana, and M. Sharma (2019), "An efficient authentication scheme for high efficiency video coding/h.265," *Multimedia Tools and Applications*, vol. 78, pp. 21245-21271, doi: 10.1007/s11042-019-7456-0.

Kundur, D., and K. Ahsan (2003), "Practical internet steganography: Data hiding in IP," *Texas workshop Security of Information Systems*.

Li, B. (2017), "3D fully convolutional network for vehicle detection in point cloud," *IEEE International Conference on Intelligent Robots and Systems*, pp. 1513-1518, doi: 10.1109/IROS.2017.8205955.

Li, M., I. Koutsopoulos, and R. Poovendran (2007), "Optimal jamming attacks and network defense policies in wireless sensor networks," *26th IEEE International Conference on Computer Communications*, pp. 1307-1315, doi: 10.1109/INFCOM.2007.155.

Li, X., C. Yu, M. Hizlan, W. Kim, and S. Park (2013), "Physical layer watermarking of direct sequence spread spectrum signals," *IEEE Military Communications Conference*, pp. 476-481, doi: 10.1109/MILCOM.2013.88.

Lin, C., and A. Sangiovanni-Vincentelli (2012), "Cyber-security for the controller area network (CAN) communication protocol," *International Conference on Cyber Security*, pp. 1-7.

Lin, S.-C., Y. Zhang, C.-H. Hsu, M. Skach, M. E. Haque, L. Tang, and J. Mars (2018),

"The architectural implications of autonomous driving," *ACM Special Interest Group on Programming Languages Notices*, 53(2), pp. 751-766, doi: 10.1145/3296957.3173191.

Liu, C., J. Chen, T. Nguyen, and M. Tomizuka (2017), "The robustly-safe automated driving system for enhanced active safety1," *SAE Technical Paper 2017-01-1406*, doi: 10.4271/2017-01-1406.

Longxiang, G., M. Sagar, L. Xuehao, and J. Yunyi (2017), "Teaching autonomous vehicles how to drive under sensing exceptions by human driving demonstrations," *SAE Technical Paper 2017-01-0070*, doi: 10.4271/2017-01-0070.

Lu, Z.-M., and S.-Z. Guo (2017), "Chapter 1- Introduction," in *Lossless Information Hiding in Images*, pp. 1-68, Massachusetts: Syngress, doi: 10.1016/B978-0-12-812006-4.00001-2.

Lucena, N. B., J. Pease, P. Yadollahpour, and S. J. Chapin (2005), "Syntax and semantics-preserving application-layer protocol steganography," in *Information Hiding*, Berlin: Springer, pp. 164-179.

Luo, X., E. Chan, and R. Chang (2008), "Tcp covert timing channels: Design and detection," *IEEE International Conference on Dependable Systems and Networks With FTCS and DCC*, pp. 420-429.

Madhavan, R., and C. Schlenoff (2003), "Moving object prediction for off-road autonomous navigation," *SPIE 5083 -Unmanned Ground Vehicle Technology V*, doi: 10.1117/12.485771.

Malik, H., K. P. Subbalakshmi, and R. Chandramouli (2008), "Nonparametric steganalysis of QIM data hiding using approximate entropy," *SPIE 6819 -Security, Forensics, Steganography, and Watermarking of Multimedia Contents X*, doi: 10.1117/12.767313.

Mercedes Benz Technologies (2018), "Carnd-mercedes-sf-utilities GitHub repository," [Online]. Available: https://github.com/udacity/CarND-Mercedes-SF-Utilities.

Miller, C., and C. Valasek (2015), "Remote exploitation of an unaltered passenger vehicle," *Black Hat USA*.

Miller, R. (2020), "Rolling zettabytes quantifying the data impact of connected cars," *Data Center Frontier,* [Online]. Available: https://datacenterfrontier.com/rolling-zettabytes-quantifying-the-data-impact-of-connected-cars, [Accessed: Jan. 21, 2020].

Mohanty, S. (2003), "Digital watermarking: A tutorial review," *ResearchGate*, [Online]. Available:https://www.researchgate.net/publication/2568630_Digital_Watermarking_A_Tutorial_Review.

Moulin, P., and R. Koetter (2005), "Data-hiding codes," *IEEE Proceedings*, 93(12), pp. 2083-2126, doi: 10.1109/JPROC.2005.859599.

Nabati, R., and H. Qi (2019), "RRPN: Radar region proposal network for object detection in autonomous vehicles," *IEEE International Conference on Image Processing*, doi: 10.1109/icip.2019.8803392.

Parah, S. A., J. A. Sheikh, N. A. Loan, and G. M. Bhat (2016), "Robust and blind watermarking technique in DCT domain using inter-block coefficient differencing," *Digital Signal Processing*, vol. 53, pp. 11-24, doi: 10.1016/j.dsp.2016.02.005.

Park, S., M.-S. Gil, H. Im, and Y.-S. Moon (2019), "Measurement noise recommendation for efficient kalman filtering over a large amount of sensor data," *Sensors*, 19(5), 1168, doi: 10.3390/s19051168.

Perera, L. D., W. S. Wijesoma, S. Challa, and M. D. Adams (2003), "Sensor bias correction in simultaneous localization and mapping," *6th International Conference on Information Fusion*, vol. 1, pp. 151-158, doi: 10.1109/ICIF.2003.177440.

Petit, J., B. Stottelaar, and M. Feiri (2015), "Remote attacks on automated vehicles sensors : Experiments on camera and LiDAR," *Black Hat Europe*, pp. 1-13.

Qin, C., P. Ji, X. Zhang, J. Dong, and J. Wang (2017), "Fragile image watermarking with pixel-wise recovery based on overlapping embedding strategy," *Signal Processing*, vol.138, pp. 280-293, doi: 10.1016/j.sigpro.2017.03.033.

Quain, J. R. (2019), "These high-tech sensors may be the key to autonomous cars," *The New York Times*, [Online]. Available: https://www.nytimes.com/2019/09/26/business/autonomous- cars-sensors.html, [Accessed: 2019].

Rigatos, G. G. (2010), "Extended kalman and particle filtering for sensor fusion in motion control of mobile robots," *Mathematics and Computers in Simulation*, 81(3), pp. 590-607, doi: 10.1016/j.matcom.2010.05.003.

SAE Ground Vehicle Standard (2018), "Taxonomy and definitions for terms related to driving automation systems for on-road motor vehicles," *SAE J3016 201806, rev. Jun. 2018*.

Sarmento, A., B. Garcia, L. Coriteac, and L. Navarenho (2017), "The challenges of the autonomous vehicle for emergent markets," *SAE Technical Paper 2017-36-0436*, doi: 10.4271/2017-01-1406.

Shin, H., Y. Son, Y. Park, Y. Kwon, and Y. Kim (2016), "Sampling race: Bypassing timing-based analog active sensor spoofing detection on analog-digital systems," *10th USENIX Workshop on Offensive Technologies*.

Shin, H., D. Kim, Y. Kwon, and Y. Kim (2017), "Illusion and dazzle: Adversarial optical channel exploits against lidars for automotive applications," in *Cryptographic hardware and embedded systems – CHES 2017*, Cham: Springer International Publishing, pp. 445-467 doi: 10.1007/978-3-319-66787-4_22.

Simon, M. K. (2002), *Probability distributions involving Gaussian random variables, a*

handbook for engineers and scientists, Boston: Springer International Publishing.

Sin-Joo Lee, and Sung-Hwan Jung (2001), "A survey of watermarking techniques applied to multimedia," *IEEE International Symposium on Industrial Electronics*, vol. 1, pp. 272-277.

Snehaprabha, N., and S. Ram (2019), *Automated parking made possible with TI mmwave radar and ultrasonic sensors*, Texas Instruments, [Online]. Available: https://www.ti.com/lit/wp/spry331a/spry331a.pdf?ts=1619895589441.

Stanely, B. (2015), "Digital data transfer is transforming the auto industry," *IBM Journey to AI Blog,* [Online]. Available: https://www.ibmbigdatahub.com/blog/digital-data-transfer-transforming- auto-industry, iBM, [Accessed: Dec. 20, 2020].

Sugawara, T., B. Cyr, S. Rampazzi, D. Genkin, and K. Fu (2020), "Light commands: Laser-based audio injection attacks on voice-controllable systems," *USENIX Security Symposium*.

TE Connectivity Ltd. (2018), "The car in the age of connectivity: Enabling car to cloud connectivity," *IEEE Spectrum*, [Online]. Available: https://spectrum.ieee.org/telecom/wireless/the-car-in-the-age-of-connectivity-enabling-car-to-cloud-connectivity, [Accessed: Dec. 20, 2020].

Tesla (2020), Model S Owner's Manual, Tesla, NA.

Texas Instruments (2014), "TI designs: TIDA-00151 automotive ultrasonic sensor interface for park assist or blind spot detection systems," [Online]. Available: https://www.ti.com/tool/TIDA-00151technicaldocuments.

Tsuji, Y. (2018), "KITTI data processing and 3D CNN for vehicle detection GitHub repository," [Online]. Available: https://github.com/yukitsuji/3D_CNN_tensorflow.

Wenjun, Z., Y. Heather, and L. Chingyung (2006), "An overview of digital watermarking," in *Multimedia security technologies for digital rights management*, pp. 167-195, Burlington: Academic Press, doi: 10.1016/B978-0-12-369476-8.X5000-3.

Woo, S., H. J. Jo, and D. H. Lee (2015), "A practical wireless attack on the connected car and security protocol for in-vehicle can," *IEEE Transactions on Intelligent Transportation Systems*, vol. 16(2), pp. 993-1006.

Woo, S., H. J. Jo, I. S. Kim, and D. H. Lee (2016), "A practical security architecture for in-vehicle can-fd," *IEEE Transactions on Intelligent Transportation Systems*, vol. 17(8), pp. 2248-2261.

Wyglinski, A. M., X. Huang, T. Padir, L. Lai, T. R. Eisenbarth, and K. Venkatasubramanian (2013), "Security of autonomous systems employing embedded computing and sensors," *IEEE Micro*, vol. 33(1), pp. 80-86, doi: 10.1109/MM.2013.18.

Xu, W., C. Yan, W. Jia, X. Ji, and J. Liu (2018), "Analyzing and enhancing the security of

ultrasonic sensors for autonomous vehicles," *IEEE Internet of Things Journal*, vol. 5(6), pp. 5015-5029.

Zhang, G., L. Kou, L. Zhang, C. Liu, Q. Da, and J. Sun (2017), "A new digital watermarking method for data integrity protection in the perception layer of IoT," *Security and Communication Networks*, pp. 1-12, doi: 10.1155/2017/3126010.

Zhou, Y., and O. Tuzel (2017), "Voxelnet: End-to-end learning for point cloud based 3D object detection," *IEEE Conference on Computer Vision and Pattern Recognition*, pp. 4490-4499, doi: 10.1109/CVPR.2018.00472.

Zolotavkin, Y., and M. Juhola (2015), "A new two-dimensional quantization method for digital image watermarking," *17th International Conference on Advanced Communication Technology*, pp. 155-160, doi: 10.1109/ICACT.2015.7224776.

Zou, Q., W. K. Chan, K. C. Gui, Q. Chen, K. Scheibert, L. Heidt, and E. Seow (2017), "The study of secure can communication for automotive applications," *17th SAE World Congress Experience*, doi: 10.4271/2017-01-1658.