SECURE VIDEO TRANSFERRING IN AUTOMOTIVE

by

Can Berk Hotamış

B.S., Mechanical Engineering, Istanbul Technical University, 2022

Submitted to the Institute for Graduate Studies in
Science and Engineering in partial fulfillment of
the requirements for the degree of
Master of Science

Graduate Program in Mechatronics Engineering

Boğaziçi University

2025

ACKNOWLEDGEMENTS

I would like to thank my advisor Assistant Professor Birkan Yilmaz who supported me throughout this project.

I would also like to thank my colleague from Daimler Truck AG, Wolfgang Zenne for fruitful discussions we had.

Finally, I am indebted to my family, friends and my beloved one, Merve, as well as our cats Poğça and Limon for their full support and encouragement especially in those times I felt the least motivated. Thank you all.

Can Berk Hotamış

ABSTRACT

SECURE VIDEO TRANSFERRING IN AUTOMOTIVE

Video monitoring became essential for the ongoing development of car systems, delivering instantaneous visual feedback for driver assistance, object detection, and safety features. As these video streams were used more extensively, the danger of security breaches with the potential to compromise the integrity and authenticity of visual data also increased. To offset such risks, digital watermarking emerged as a tool, with the capability of embedding hidden authentication data in video streams without affecting perceptual quality.

This work evaluated the performance of four watermarking schemes—Base Effective Algorithm, Sequenced Watermarking, Multi-Subband Watermarking, and HMAC-Enhanced Watermarking—under varying attack scenarios, including additive Gaussian noise, compression, cropping, and image swapping. Using objective quality metrics such as Peak Signal-to-Noise Ratio (PSNR) and Normalized Cross-Correlation (NCC), we quantified visual quality versus watermark robustness trade-offs. Experiments were conducted under both light and heavy Gaussian noise scenarios to simulate real-world tampering.

It was concluded that under low attack levels, visual quality was best preserved by the Base Algorithm, whereas Multi-Subband Watermarking performed better under more aggressive manipulation conditions. Sequenced Watermarking proved promising in detecting frame-level forgeries, and HMAC-based embedding provided cryptographic security. These findings provided a strategic basis for selecting watermarking methods based on security requirements for video surveillance in automotive systems.

ÖZET

OTOMOTİVDE GÜVENLİ VİDEO İLETİMİ

Video izleme, sürücülere destek, nesne algılama ve güvenlik özellikleri için anlık görsel geri bildirimi sağlayarak gelişen araç sistemlerinin ilerlemesinde önemli hale gelmiştir. Video akışlarının daha yaygın kullanılmasıyla birlikte, görsel verilerin bütünlüğünü ve doğruluğunu tehlikeye atabilecek güvenlik ihlalleri riski de artmaktadır. Bu tür riskleri azaltmak amacıyla, görsel kaliteyi bozmadan video akışlarına gizli kimlik doğrulama verisi yerleştirme yeteneğine sahip dijital filigranlama bir çözüm olarak öne çıkmaktadır.

Bu çalışmada, dört farklı filigranlama algoritmasının—Temel Etkili Algoritma, Sıralı Filigranlama, Çok Alt Bant Filigranlama ve HMAC tabanlı Filigranlama performansları, Gauss gürültüsü ekleme, filtreleme, kırpma ve görüntü değiştirme gibi çeşitli saldırı senaryoları altında değerlendirildi. Tepe Sinyal-Gürültü Oranı (PSNR) ve Normalleştirilmiş Çapraz Korelasyon (NCC) gibi nesnel kalite ölçütleri kullanılarak, görsel kalite ile filigran sağlamlığı arasındaki denge değerlendirildi. Gerçekçi müdahale senaryolarını simüle etmek üzere hem hafif hem de yoğun Gauss gürültüsü durumlarında deneyler gerçekleştirildi.

Elde edilen sonuçlara göre, düşük saldırı seviyelerinde en yüksek görsel kalite Temel Etkili Algoritma ile sağlandı; ancak daha güçlü saldırılar karşısında en iyi performans Çoklu Alt Bant Filigranlaması ile elde edildi. Sıralı Filigranlama, video karesi düzeyindeki sahteciliklerin tespitinde umut vadederken; HMAC tabanlı yöntem ise kriptografik güvenlik katkısı sunmaktadır. Bu bulgular, otomotiv sistemlerindeki video gözetimi için güvenlik gereksinimlerine uygun filigranlama algoritmalarının stratejik olarak seçilmesine olanak sağlamaktadır.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

LIST OF SYMBOLS

$C(t)$	Scaling Factors for DCT
$corr2(A, B)$	2D Correlation Coefficient
$\cos(t)$	Cosine Function
$\log_{10} a$	Logarithm with Base 10
μ_X	Mean
π	Pi
$\phi(t)$	Scaling Function
$\psi(t)$	Mother Wavelet Function
σ	Standard Deviation
$\sum$	Summation

LIST OF ACRONYMS/ABBREVIATIONS

ADAS	Advanced Driving Assistance Systems
APIX	Automotive Pixel Link
APVSS	ASIL Prepared Video Safety System
CAL	Cybersecurity Assurance Level
CAN	Controller Area Network
CIA	Confidentiality, Integrity, and Availability
CML	Current Mode Logic
CMS	Camera Monitor System
CRC	Cyclic Redundancy Check
DCT	Discrete Cosine Transform
DFT	Discrete Fourier Transform
DSC	Display Stream Compression
DWT	Discrete Wavelet Transform
E/E	Electrical and Electronic
EMC	Electromagnetic Compatibility
EMI	Electromagnetic Interference
FPD-Link	Flat Panel Display Link
GMSL	Gigabit Multimedia Serial Link
HDCP	High-Bandwidth Digital Content Protection
HMAC	Hash-based Message Authentication Code
HiL	Hardware-in-the-loop
ISO	International Organization for Standardization
LIN	Local Interconnect Network
LSB	Least Significant Bit
LVDS	Low-Voltage Differential Signaling
MAC	Message Authentication Code
MSE	Mean Squared Error
NCC	Normalized Cross-Correlation
OEM	Original Equipment Manufacturer

PAM	Pulse-Amplitude Modulation
PRN	Pseudo-Random Number
PSNR	Peak Signal to Noise Ratio
RGB	Red Green Blue
SERDES	Serializer Deserializer
STP	Shielded Twisted Pair
SVD	Singular Value Decomposition
TARA	Threat Analysis and Risk Assessment
UART	Universal Asynchronous Receiver / Transmitter

1. INTRODUCTION

1.1. Problem Definition and Motivation

Video monitoring in the automotive industry plays a significant role by providing real-time information to the driver. With the increase in the use of autonomous driving and Advanced Driver Assistance Systems (ADAS), video monitoring technologies have become crucial. One of the key ecosystems in automotive is the Camera Monitor System (CMS), which typically consists of a camera, an Electronic Control Unit, and a display unit; all connected together with the help of different solutions of video links. CMS must be able to meet the security and safety requirements of automobiles. One of the major issues today is the lack of strong cybersecurity measures for automotive CMS. For instance, currently there is no process that is being widely used to authenticate the video streams; the system, therefore, is prone to fault scenarios and attacks. A malicious entity is able to exploit the outside interfaces to exploit the video data, thereby having the ability to cause safety issues. The gap has to be filled by pugging the loophole to enable the CMS to become reliable and safe in the present-day automobile systems.

One of the approaches used by automotive camera interfaces is Serializer-Deserializer (SerDes) communication protocol for its very high data rate and low latency. In SerDes links, communication is performed either over a Shielded Twisted Pair (STP) or coaxial cable in the physical layer. 3 broad categories of SerDes video links are present.

The Flat Panel Display Link (FPD-Link) by Texas Instruments, currently in its third generation, offers a forward channel bandwidth exceeding 25 Gbps with a simultaneous low-speed back channel. Latency must also be minimal in the case of autonomous vehicles, and compression and decompression processing needs must be choreographed carefully. Texas Instruments processors provide High-Bandwidth Digital Content Protection (HDCP), with cryptographic techniques used to safeguard data transfer [5].

APIX, developed by Inova Semiconductors, serves as a unified interface for both display and remote digital camera applications. It utilizes Current Mode Logic (CML) technology to serialize and transmit data, while also incorporate HDCP in certain processors [6].

Gigabit Multimedia Serial Link (GMSL) developed by Maxim Integrated is the most sought-after among OEMs. GMSL 2 allows operation in a forward channel data rate of 3 Gbps or 6 Gbps and in the backward channel a data rate of 187 Mbps. Maxim Integrated provides HDCP and watermarking features in some of its processors, such as the MAX9266 and MAX86140. Even though there are different types of links, SerDes video links allow the integration of different links into each other.

CMS should comply with independent automotive standards like ISO 16505 that includes performance requirements for CMS extensively utilized by OEMs. The CMS should also be protected from cyber attacks, for which ISO 21434 provides directions and principles for automotive Cybersecurity.

Several studies have explored various aspects of CMS. Gutzmer highlights the growing complexity of camera interface specifications driven by technical advancements and varied industry requirements [1]. His work punctuates standardization to promote affordability in multiple automotive applications. Terzis addresses the replacement of auto mirrors by CMS to improve driver sight, aerodynamics, and fuel efficiency [7]. Addressing challenges, he emphasizes CMS alignment with ISO 16505 for rear-view mirror replacement. Ensuring CMS functionality equivalent to traditional mirrors, the discussion covers real-time behavior, system resolution, and display parameters [7].

Witt and Bauer propose a robust real-time video watermarking technique for frozen frame detection in video systems for CMS functional safety. Their processor buffering-resistant, video compression-resistant, and common image processing operations-resistant technique achieves robustness and low false detection probability. By distributing the watermark's impact temporally, this approach ensures the watermark remains below the perceptual threshold, as verified by objective methods [8].

Axmann et al. contribute to CMS safety by focusing on advanced monitoring of panel electronics and display output. Their approach, ASIL Prepared Video Safety System (APVSS), ensures safety-verified Red Green Blue (RGB) data is supplied to panel electronics. In case of failure, their system facilitates a safe state by deactivating the panel. They also mention graceful degradation, pointing out the necessity of maintaining a safe but marginally degraded image for improved customer experience. Their approach is made effective by presenting a working prototype and strict tests, which once again proves the significance of 'light-to-light' supervision in CMS [9].

Watermarking, as a security measure, can be implemented through several techniques, including frequency domain watermarking, which uses transforms such as Discrete Cosine Transform (DCT), Discrete Wavelet Transform (DWT), and Singular Value Decomposition (SVD). These techniques provide a much better cover such that even human vision cannot detect any noticeable difference and they ensure high robustness against typical image processing operations. However, spatial watermarking has certain robustness and imperceptibility limitations which make it less suitable for video transmission and integrity verification applications.

Rahinj describes that digital watermarking is the act of embedding information within digital content—e.g., images, audio, video, or relational databases to carry out several security tasks like ownership assertion, tampering detection, and traitor tracing. The phrase "digital watermark" was first coined by Andrew Tirkel and Charles Osborne in 1992, with whom Gerard Rankin also demonstrated the first successful use of a spread-spectrum steganographic watermark. In Rahinj's opinion, watermarking for digital data is a vital tool for data integrity and intellectual property, particularly in large-scale data exchange and big data environments such as social networks. It is highlighted as being more difficult and variable than traditional security mechanisms, with the capability to safeguard not only multimedia but even structured data such as relational databases [10].

Melman and Evsutin provide a systematic review of robust watermarking schemes, specifically how they protect against targeted attacks where they also introduce the basics of digital watermarking. They provide a four-level classification of attacks based on attacker intention, target, action features, and implementation method. They explore watermark resilience to various challenges like image processing, geometry manipulation, print-scan operations, collusion, and ambiguity attacks. Their paper gives a detailed summary of current countermeasures and outlines future directions for research in enhancing watermarking security. They also give a definition of Digital watermarking as a technique of information embedding in digital images to satisfy various security needs such as copyright, authenticity, and integrity verification. According to their strength, watermarks are categorized as fragile, semi-fragile, or robust. Fragile watermarks are destroyed on any modification and are used for data authenticity and integrity (e.g., medical images). Semi-fragile watermarks resist some processing distortions (e.g., compression) but can detect content tampering. Robust watermarks resist a set of processing operations and are primarily used for copyright protection and content tracking [11].

Regarding cybersecurity in watermarking, Agbaje et al. assert that cybersecurity extends beyond traditional IT security, protecting systems and data from diverse cyber threats. Their paper explores cyber watermarking, employing digital watermarking to address information theft in cyberspace, using a literature review and case study methodology [12].

Wu et al. proposed a lightweight digital watermarking approach for security enhancement of in-vehicle Controller Area Network (CAN) systems without modification of the standard CAN protocol. They employed a string-based watermark compressed using Huffman coding, instead of traditional message authentication codes (MACs) which are bandwidth-consuming and computationally intensive. The compressed watermark was split and embedded byte-by-byte in the data fields of CAN messages. At the receiving end, the watermark is reconstructed and verified for authenticity, allowing effective detection of message injection or tampering attacks. The method supports dynamic regeneration of the watermark using a linear congruential generator, foiling

replay attacks while allowing flexibility. Implemented on a real CAN bus testbed, the proposed solution achieved satisfactory detection accuracy with very low latency (1 ms) while adhering to the timing and format constraints of the CAN protocol [13].

Nonetheless, the application of watermarking to automotive cybersecurity is mainly an unexplored field. Though it has many potential applications in improving the security level of automotive systems, most notably CMS and other video transmission systems research on how watermarking can be performed and efficient for automotive cybersecurity is missing.

1.2. Aim and the Contributions of the Study

The aim of this study is to investigate the complexity of SerDes video interfaces, examine existing security models of automobile data exchange and perform watermarking combination testing and analysis. This thesis' contributions can be listed as:

- Analysis of SerDes video interfaces
- Evaluation of classical security schemes
- Implementation of frequency domain watermarking for security
- Evaluation of watermarking performance under different fault scenarios
- Evaluation of watermarking performance under different attack types
- Enhancement of CMS security with watermarking combined with different mechanisms

2. CAMERA MONITOR SYSTEMS AND VIDEO INTERFACES

2.1. System Architecture

The automotive industry has witnessed tremendous advancements by integrating increasingly sophisticated electronic subsystems such as cameras taking over conventional physical mirrors. Technologies that have been originally introduced into electronic systems as luxury systems are now ubiquitous after regulatory requirements. An example is the Camera Monitoring System. As an instance, new cars in the United States have been required by law to include rear-view cameras since 2018 [14].

The general structure of CMS encompasses a camera as the input device, a control unit, and a display. The video information is captured by the camera, transmitted to the control unit, where the signal is processed to present the best image as also demonstrated in Figure 2.1. The video is then displayed in real time on a display unit to provide enhanced driver vision. For instance, the Mirror-Cam systems use A-pillar-mounted displays on trucks instead of side mirrors, while rear-view cameras use head-up displays for better driver access. One-way video transmission from camera to display is usually the norm.

2.2. Communication Topologies and Video Interfaces

Communication within a vehicle for CMS can use various topologies. Traditional in-vehicle networks such as CAN and LIN are based on a bus topology, which allows efficient data sharing. Even though CAN and LIN are widely deployed and relatively inexpensive, the use of shared access and arbitration mechanisms in these traditional networks restricts the available bandwidth, rendering them unsuitable for high-bandwidth applications such as video [1].

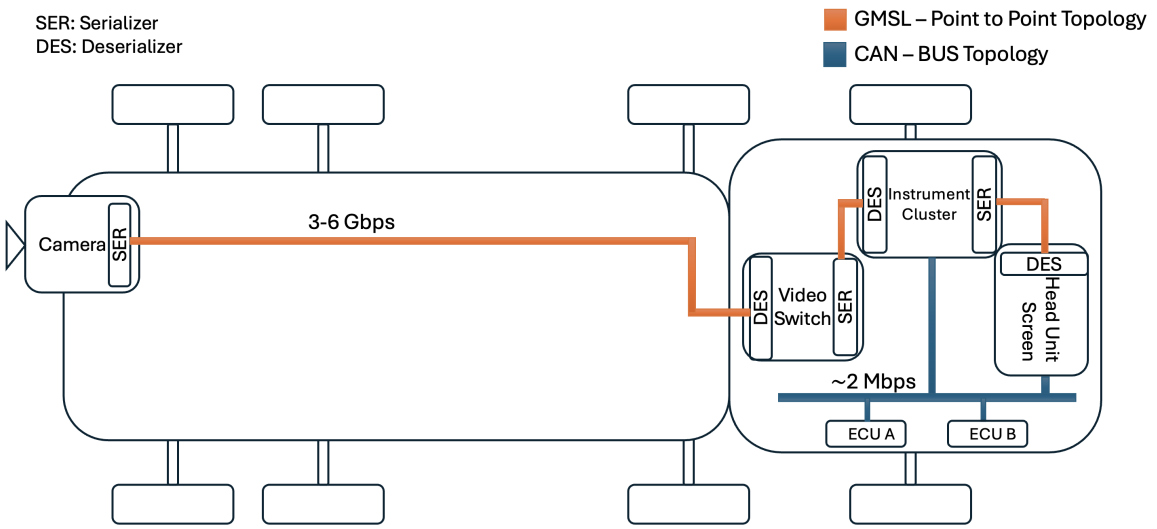
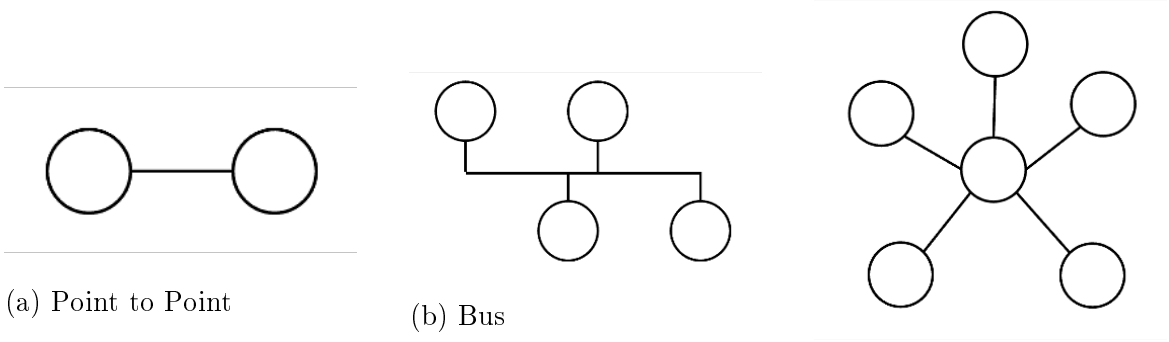


Figure 2.1. Simple Architecture of CMS with GMSL and CAN-BUS links.

In practice, point-to-point topology is commonly used for that purpose in most CMSs by establishing a direct link from one component to another, such as from camera to display. A point-to-point channel offers high-bandwidth and low-latency transmission that is, for the given application, required by video signals to be displayed in real time without delay. A point-to-point topology has, therefore, been one of the more popular foundations in most of today's automotive video interfaces.

Star topology consists of several point-to-point links to a central node. All data traffic between two nodes has to pass through the central node [1]. Figure 2.2 shows the simple structures of different network topologies. Specifically, Figure 2.2 (a) shows the Point to Point topology, Figure 2.2 (b) shows the Bus topology, and Figure 2.2 (c) shows the Star topology.

Ethernet provides robust data transmission protocols, transceivers, and security mechanisms. However, as it is a video link, it is expensive and is utilized mainly in niche applications rather than general use for CMS [15]. Topologies like bus and star are utilized by Ethernet.



(a) Point to Point

(b) Bus

(c) Star

Figure 2.2. Different Topologies (adapted from [1])

Analog cameras were more often utilized due to their Electromagnetic Compatibility (EMC) compliance and compatibility with long cables. However, when digital displays were demanded, analog became less practical since it meant an additional step of digitization, which added cost and complexity [1]. Overall Table 2.1 shows a comparison for utilizing Analog cameras, Ethernet and Point to Point topology for CMSs. SerDes interfaces which utilize point to point topology are widely utilized in automotive video applications due to their flexibility and high data rates. They also operate over a variety of cable types such as coaxial cable and twisted-pair cable, and encapsulated video transmission support to offer interoperability between different video components. Most SerDes implementations normally use LVDS for low power consumption and CML to meet high-speed requirements.

It is a trade-off between cost and bandwidth and signal integrity, and each has its own advantages since manufacturers can choose interfaces to optimally suit their particular needs for CMS. Some are more software-compatible than others, although most suppliers, including Texas Instruments and Analog Devices, produce products expecting use with SerDes video interfaces. The 3 SerDes links are Flat Panel Display Link, automotive Pixel Link and Gigabit Multimedia Serial Link.

Table 2.1. Comparison of CMS Communication Methods

Aspect	*Ethernet Based*	*Analog Camera Based*	*Point to Point Topology Based*
Bandwidth	Very high (supports HD video)	Low (limited by analog signal capacity)	High (dedicated channel enables fast transfer)
Topology	Primarily Star or Bus	Typically Point-to-Point	Point-to-Point (dedicated camera-to-display link)
Cost	High (due to hardware, shielding, protocol stack)	High (due to adaptation to digital systems)	Moderate (requires a link per connection)
Complexity	High (protocol stack, routing)	Low (but increases when digitization is needed)	Low to moderate (direct wiring)

2.3. Flat Panel Display Link

Flat Panel Display Link (FPD-Link) was first developed by Texas Instruments. Now in its third generation, FPD-Link III, it provides a forward channel bandwidth exceeding 25 Gbps and a simultaneous low-speed back channel. The back channel, operating at versatile speeds, allows for functions like I2C bus transport and GPIO control without interrupting the video flow. In autonomous vehicles, minimizing latency is crucial, and the additional processing for image compression and decompression contributes to this consideration [16]. It has a link speed up to 3 GB/s [1].

Regarding security, some of Texas Instruments processors like DS90UH949-Q1 provide High-Bandwidth Digital Content Protection (HDCP). HDCP uses cryptography to secure the data transmission. More information on HDCP will be provided.

2.4. Automotive Pixel Link APIX™

APIX was developed by Inova Semiconductors. APIX architecture serves as a unified interface for either a display or a remote digital camera solution. It uses CML technology to serialize and transmit data, enabling high-speed transmission at distances of up to +15 m (1 Gbit/s mode) and up to +40 m (500 Mbit/s mode). The APIX link is structured with three independent channels, facilitating full-duplex transmission in both directions. The upstream sideband that has a speed up to 20 Mbit/s, downstream sideband channel which is up to 26 Mbit/s and a high-speed downstream pixel channel with speed up to 1 Gbit/s [6]. For 12 meters, it has a link speed up to 3 GB/s and for 40 meters it has a link speed up to 500 Mb/s [1]. Regarding security, Inova Semiconductors also provides HDCP functionality in some of its processors, such as the INAP596TAQ.

2.5. Gigabit Multimedia Serial Link

Gigabit Multimedia Serial Link (GMSL) was developed by Maxim Integrated. One of the most common versions that is used by many OEMs, GMSL 2 facilitates an SPI tunnel with data rates of 25 Mb/s or 50 Mb/s, supporting two UART tunnels with a data rate of up to 5 Mb/s. Notably, a high transmit frequency enables the use of smaller size inductors for power delivery over the cable. The forward channel employs PAM2 modulation to achieve these functionalities [17].

For 15 meters, it has a link speed which is up to 3.1 GB/s [1]. Maxim Integrated provides in some of its processors like MAX9266. Maxim Integrated also provides in some of its processors like MAX86140, the watermarking feature. Watermarking will be examined thoroughly in the upcoming chapter.

An example video transmission for GMSL2 is shown in Figure 2.3. Between the source and the sink there are GMSL2 serializer and deserializer with a connection of GMSL2 Channel. The capacities of the channels are also depicted.

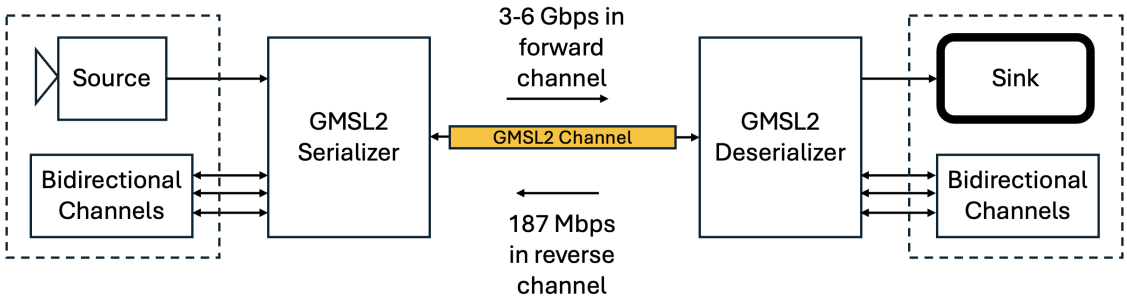


Figure 2.3. GMSL2 Block Diagram (adapted from [2])

3. CYBERSECURITY AND SECURING THE VIDEO TRANSFER

With the increase of digitization within the automotive industry so did there grow concern and relevance to cybersecurity, especially as cars will bring together technologies like in-vehicle and vehicle-to-vehicle communications and autonomous driving systems. Strong cybersecurity measures are of paramount importance for the protection of data integrity, ensuring the triad CIA: Confidentiality, Integrity, and Availability with regard to vehicle and network security [18].

ISO 21434 sets "security by design" as the principle for how the vehicle is to be secured against cyber attacks. It lists specific security measures to protect against fault scenarios and attacks. Different cryptographic methods, such as symmetric and asymmetric key cryptography and hash functions, underpin the requirements. For multimedia data, like video streams, special methods such as HMAC and digital watermarking may be used to fulfill the purpose.

ISO 21434 provides a comprehensive framework to address cybersecurity risks in the engineering of electrical and electronic (E/E) systems of road vehicles, from the concept phase through decommissioning. The norm highlights the importance of cybersecurity and risk management policies and promotes a culture of cybersecurity. It elaborates on the necessity of establishing and maintaining cybersecurity governance, project-specific cybersecurity operations, and the delineation of supplier and customer responsibilities for this. Other continual cybersecurity activities include, but are not limited to, monitoring, event evaluation, and vulnerability management to provide confidence that risks are continuously assessed and acted upon. In the concept phase, the product is designed, cybersecurity requirements are defined, and a cybersecurity concept is derived from Threat Analysis and Risk Assessment (TARA) methods. Product development entails the definition of cybersecurity requirements, along with integration and verification activities that ensue. Validation is centered on testing and reviewing

measures to make certain objectives will be reached. This standard also covers production, operations, maintenance, and decommissioning with respect to cybersecurity aspects. Among key concepts, cybersecurity risk management, establishment of a cybersecurity culture, TARA approaches, and cybersecurity assurance levels (CALs) are to be viewed. A cyber-secure system must adhere to an organization's processes, in accordance with established procedures: establishment of policies, definition and confirmation of cybersecurity goals, management of vulnerabilities. Validation would involve specifying the criteria, creating a master plan, and performing tasks such as static and dynamic analysis, rigorous testing, result evaluation, and extensive documentation, all in an attempt to verify that cybersecurity goals—such as data integrity—have been properly satisfied.

Cybersecurity goals refer to requirements that are necessary to provide protection for identified assets against threat scenarios. These are outcomes of a careful risk evaluation process and should be customized based on system characteristics. Some examples of cybersecurity goals include:

- *Confidentiality:* Guarding confidential information, such as personal information and cryptographic keys, from being disclosed to the wrong people.
- *Integrity:* Signals and data - including others such as control commands and firmware downloads - are not modified or altered by the wrong individuals.
- *Availability:* Key functions, including braking, steering, and others, are accessible when needed even during cyber threats.
- *Authentication:* Unauthorized access or control to the system, e.g., secure boot and user authentication.
- *Non-repudiation:* Demonstrating that events or things can be shown to have occurred, i.e., excluding denial by the parties (e.g., auditing and logging).

These goals are not universal but are specific to each system based on its unique risk profile, ensuring effective protection against threats [3].

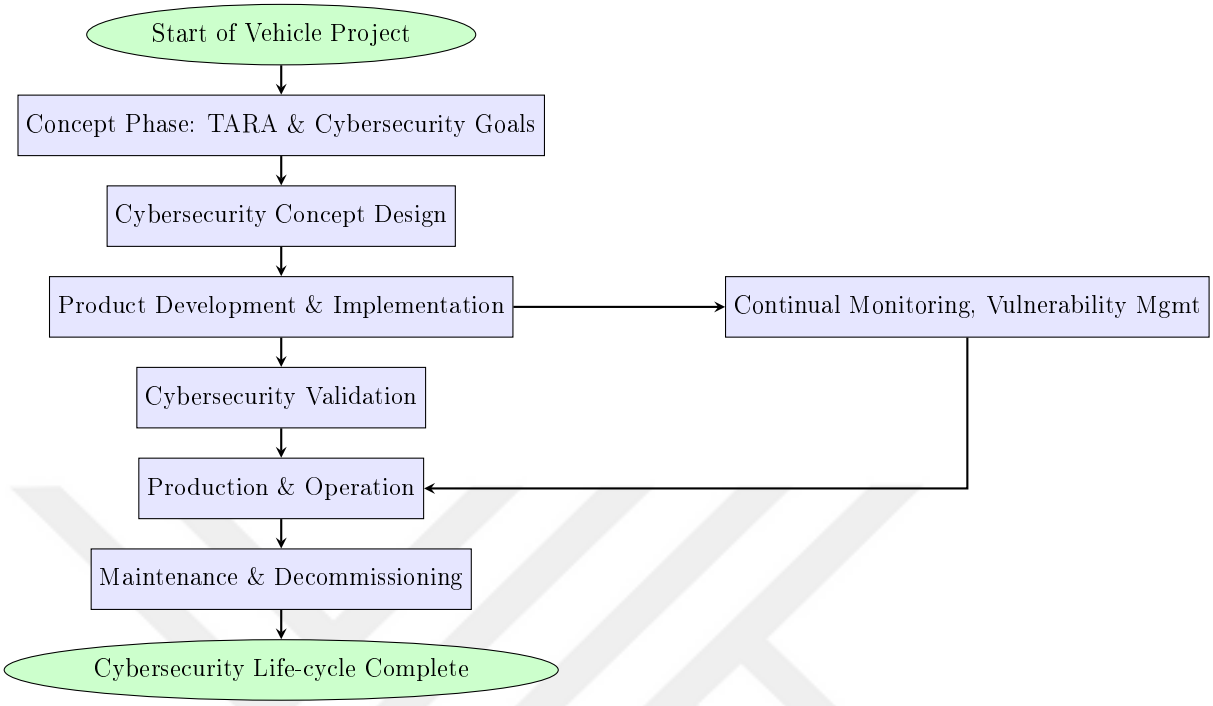


Figure 3.1. Cybersecurity Framework for Automotive (adapted from [3])

Figure 3.1 shows the Framework for Cybersecurity application in automotive according to ISO 21434. Important phases are listed between the start of vehicle project and the completion of the cybersecurity life-cycle.

3.1. Cybersecurity Methods

Cybersecurity is methods to protect information from unauthorized access. Common methods to authenticate information integrity are checksums, Cyclical Redundancy Check (CRC), and cryptographic hash functions.

Checksums have only basic data error detection without strong security properties concerning intentional manipulation.

CRC is a powerful technique widely used in auto; it accommodates rapid error detection but is vulnerable to malicious attack.

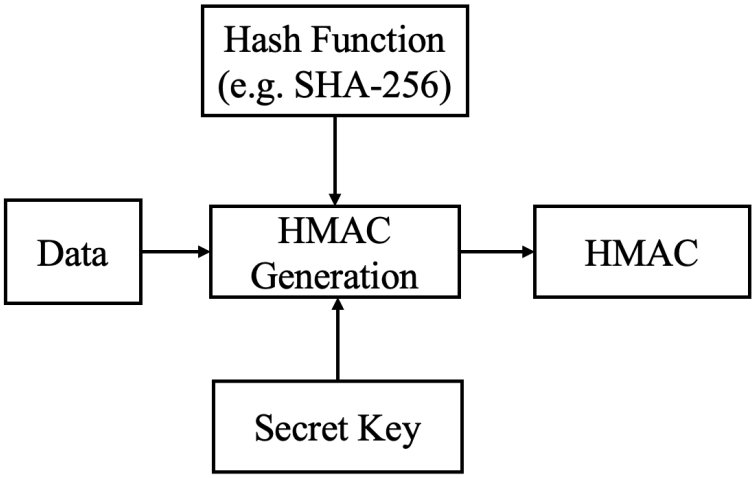


Figure 3.2. HMAC Generation Block Diagram

Hash Functions provide a unique value from input data, and these functions are in wide use in cybersecurity. A characteristic of the cryptographic hash function adds security, as slight changes in the input data will produce completely different hash outputs.

Traditional methods have their own limitations regarding image and video data. In fact, common errors in multimedia content, such as corruptions in bits, may reduce their effectiveness. Furthermore, the efficacy of encryption protection is lost when content is decrypted. Digital watermarking is thus able to offer verification even after decryption and analog conversion. The most important application here is automotive video systems, where the verification of video streams at the destination needs to be secure.

3.2. HMAC (Hash-Based Message Authentication Code)

HMAC is a method of deriving a cryptographic hash from a message and a secret key. This provides a secure hash that can be used to ascertain the integrity and authenticity of a message. In HMAC, tamper resistance comes from the fact that only one with the correct key will be able to generate the hash. In video transmission for automotive use, HMAC-SHA256 is widely used. Usually, this will compute an HMAC

over every frame or segment of the video, which is checked at the destination for data integrity. Figure 3.2 shows the Block Diagram for HMAC Generation

The major features of HMAC:

- *Integrity Protection:* HMAC also hashes the message with a secret key so that the content is not modified.
- *Authentication:* The secret key allows only the key owner to authenticate or modify the data concerned, in effect adding an additional layer of authenticity [19].

3.3. Digital Watermarking

Digital watermarking is a method used in addition to encryption for offering data security and authenticity. By embedding a secret "signature" within video or image data, watermarking allows verification after transmission.

3.3.1. Non-Visible Watermarking

In automotive applications, the use of non-visible watermarking is most preferred to not to disrupt the video.

Non-visible watermarking is inserting information which cannot be detected by the viewer's eye but system components can detect it. It resembles something similar to steganography; however, it concentrates on the purpose of ensuring authenticity rather than hiding data from unauthorized viewers.

3.3.2. Watermarking Domains

Digital watermarking techniques are broadly classified into two classes based on its domain: spatial domain and frequency domain.

- *Spatial Domain Watermarking:* It is the process of embedding the watermark directly into the pixels of the image or video. One of the most popular techniques in this category employs Least Significant Bit (LSB) watermarking, where the least significant bits of the selected pixels are altered. This technique is highly imperceptible but also susceptible to attacks.
- *Frequency Domain Watermarking:* Frequency domain is the preliminary step towards converting the original data. Therefore, a watermark is more resilient to affine transformations like resizing, cropping, and rotation. One of the popular frequency domain techniques for digital watermarking approaches in the context of multimedia applications is the Discrete Wavelet Transform.

Generally used frequency domain transfers are Discrete Cosine Transformation and Discrete Wavelet Transformation.

3.3.3. Discrete Cosine Transform

The Discrete Cosine Transform (DCT) is widely used in signal processing and image compression, converting data points or images from spatial to frequency domains using real cosine functions—distinct from the complex exponentials of the Discrete Fourier Transform (DFT). Common in compression methods like JPEG, the DCT concentrates signal energy into a few coefficients, enabling compact representation with minimal loss of visual quality at high compression ratios. Its one-dimensional formula, expressed with cosine functions, has a reversible nature, allowing approximate reconstruction of original data from coefficients [20]. Similar to DWT, DCT can also be used to decompose the different dimensional data. The two-dimensional DCT can be formulated as below with,

$$F(u,v) = \frac{2}{\sqrt{MN}} C(u)C(v) \sum_{i=0}^{M-1} \sum_{j=0}^{N-1} f(i,j) \cos\left[\frac{(2i+1)u\pi}{2M}\right] \cos\left[\frac{(2j+1)v\pi}{2N}\right]. \quad (3.1)$$

where M and N are the dimensions of the image, i is the row index, j is the column index of the image.

DCT can be described as below where $C(u)$ and $C(v)$ are scaling factors for the two-dimensional,

$$C(u) = \begin{cases} \frac{1}{\sqrt{2}} & \text{if } u = 0 \\ 1 & \text{otherwise} \end{cases}, \quad C(v) = \begin{cases} \frac{1}{\sqrt{2}} & \text{if } v = 0 \\ 1 & \text{otherwise,} \end{cases} \tag{3.2}$$

where $f(x, y)$ represents the pixel value at spatial coordinates (x, y) in the original image, the inverse two-dimensional DCT can be described as below,

$$F(u, v) = \frac{2}{MN} \sum_{x=0}^{M-1} \sum_{y=0}^{N-1} C(x)C(y)f(x, y) \cos\left[\frac{(2u+1)x\pi}{2M}\right] \cos\left[\frac{(2v+1)y\pi}{2N}\right]. \tag{3.3}$$

3.3.4. Discrete Wavelet Transform (DWT)

The Discrete Wavelet Transform (DWT) plays a crucial role in this study's video watermarking algorithm. In particular, the watermark is embedded in the Cb channel of each video frame using the LL subband obtained through a two-level biorthogonal DWT. This approach is selected for its ability to provide both frequency and spatial domain localization, thereby supporting imperceptible and robust watermark embedding.

Wavelet analysis decomposes a signal into scaled and translated versions of a basic waveform, known as the mother wavelet. Unlike the Fourier Transform, which analyzes frequency content without spatial localization, the wavelet transform is capable of localizing information in both time and frequency domains. This makes it especially well-suited for image and video processing applications.

Wavelet functions, represented generally by $\psi(t)$, generate a family of functions through scaling and shifting. These are used to analyze signals at different resolutions,

$$\int_{-\infty}^{\infty} \psi(t), dt = 0. \tag{3.4}$$

In practice, the Discrete Wavelet Transform (DWT) is preferred over the Continuous Wavelet Transform (CWT) for computational efficiency. DWT discretizes both time and scale, thus enabling fast implementations suitable for image and video wa-

termarking. This work utilizes the 2-dimensional DWT (2D-DWT), which is ideal for grayscale or individual color channels of video frames. In this study, we specifically operate on the Cb channel of YCbCr color space, since chrominance channels are perceptually less sensitive, allowing for watermark embedding without significant visual degradation. Different types of wavelets can be used for decomposition. This study adopts the `bior6.8` biorthogonal wavelet, which offers symmetric filters and perfect reconstruction properties—important advantages for maintaining visual quality and enabling accurate watermark recovery. Orthogonal wavelets like Haar and Daubechies are energy-preserving but lack symmetry, making them less suitable for preserving image edges. The biorthogonal wavelet used here enables better imperceptibility, especially in the LL subband, which is critical when dealing with visual data such as video frames [21].

DWT decomposes an image into four frequency subbands: LL (approximation), LH (horizontal), HL (vertical), and HH (diagonal). These are obtained through sequential filtering and downsampling along rows and columns. The mathematical representation of these 2D-DWT subbands is given as,

$$LL = \phi(x, y) = \phi(x)\phi(y) \tag{3.5}$$

$$LH = \psi_H(x, y) = \psi(x)\phi(y) \tag{3.6}$$

$$HL = \psi_V(x, y) = \phi(x)\psi(y) \tag{3.7}$$

$$HH = \psi_D(x, y) = \psi(x)\psi(y) \tag{3.8}$$

where $\phi(x)$ denotes the scaling function and $\psi(x)$ the wavelet function [22]. In MATLAB's implementation, these subbands are accessible via functions such as `appcoef2` (LL subband), and `detcoef2` (LH, HL, HH bands). These correspond to the variables `cA1, cH1, cV1, cD1` in the code provided in Appendix A.1. For instance, the following lines from the code illustrate how the LL subband is extracted and manipulated:

```
[C1, S1] = wavedec2(coverimage, N, wavetype);
cA1 = appcoef2(C1, S1, wavetype, N); % LL subband
```

This LL subband (`cA1`) is where the watermark is embedded by modulating it with a pseudorandom sequence. As LL contains the most perceptually important information, it ensures high robustness to signal processing attacks such as cropping, noise, or compression. The watermark insertion in LL subband is designed to be imperceptible but detectable, which is key for applications involving integrity verification and tamper detection.

After embedding, the inverse DWT is used to reconstruct the watermarked image:

```
watermarked_Cb = waverec2(C1, S1, wavetype)
```

The reconstruction preserves the watermark invisibly within the Cb channel of the frame, and thus within the final watermarked video stream. This allows extraction and verification even after common video transformations.

In summary, DWT is used in the proposed algorithm because:

- It provides multi-resolution analysis suitable for hierarchical watermark embedding.
- The LL subband retains critical image information, enabling robust watermarking.
- Biorthogonal wavelets like `bior6.8` ensure minimal distortion due to their symmetry and perfect reconstruction.
- The decomposition fits naturally into framewise video watermarking pipelines.

The diagram in Figure 3.3 illustrates the spatial arrangement of the subbands obtained after one level of 2D-DWT decomposition, which is recursively applied to LL for further granularity in multi-level DWT. Further practical implementation details can be found in Appendix A.1, where the code listings for DWT-based watermark embedding and extraction processes are fully explained.

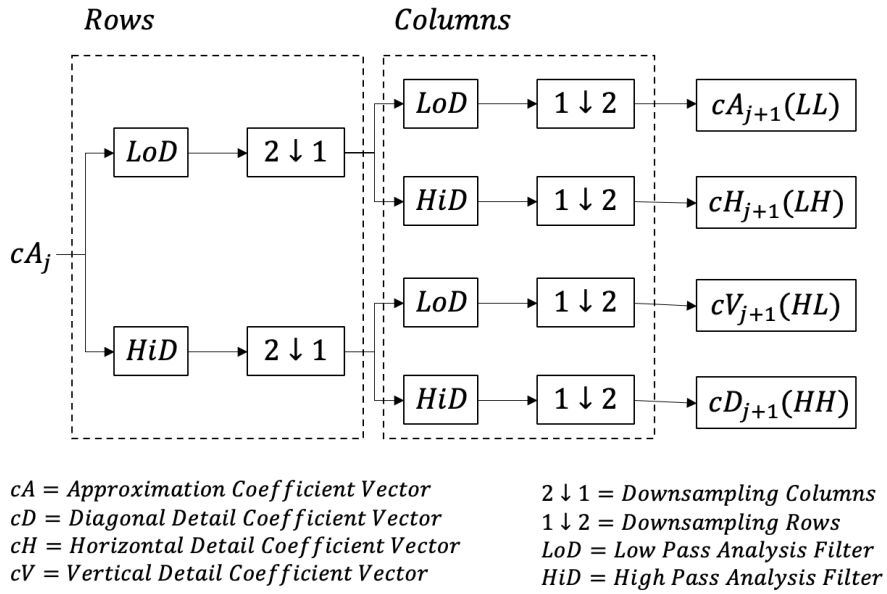


Figure 3.3. DWT Subband Decomposition Structure (adapted from [4])

4. ALGORITHMS

4.1. Introduction

In this thesis, combinations of watermarking are used to create videos with embedded watermarks. The first method uses basic DWT watermarking, hence it is not combined with any other security method. Building on that, three methods are introduced: one uses a sequence of watermarks over multiple frames for dropped-frame detection, another integrates HMAC for added integrity, and the third embeds the watermark across multiple subbands to decrease false positives/negatives.

These methods attempt to add robustness and security to watermarking at the expense of slight degradation in the quality of the image. Watermarking will be applied in the serializer within the camera unit. After the video is transmitted to the deserializer of the display, the extraction of the watermark is done. A threshold for NCC (Normalized Correlation Coefficient) is to be set in the deserializer of the display above which shows the video as being not tampered with. Embedding and extracting is done using a PRN (Pseudo-Random Number) process which is initiated by a seed. The additional PRN sequence step adds extra security to the process which is inspired by Cox et al. [23]. A description for the algorithms proposed to be used with their security aspects can be seen in Table 4.1.

4.2. Watermark Embedding and Extraction

In this chapter for each algorithm embedding and extraction procedures are given.

4.2.1. Base Effective Algorithm - Single Subband Watermarking (Shown in Figure 4.1)

Base effective algorithm primarily achieves Integrity, since it ensures video data is not tampered with in a stealthy manner.

Table 4.1. Security Feature Matrix of Proposed Algorithms

Watermarking Method	*Integrity*	*Authentication*	*Non-repudiation*	*Availability*	*Drop Frame Detection*
Base Effective	✓				
Sequenced Watermarking	✓				✓
HMAC Embedded Watermarking	✓	✓	✓		
Multi-Subband Watermarking	✓			✓	

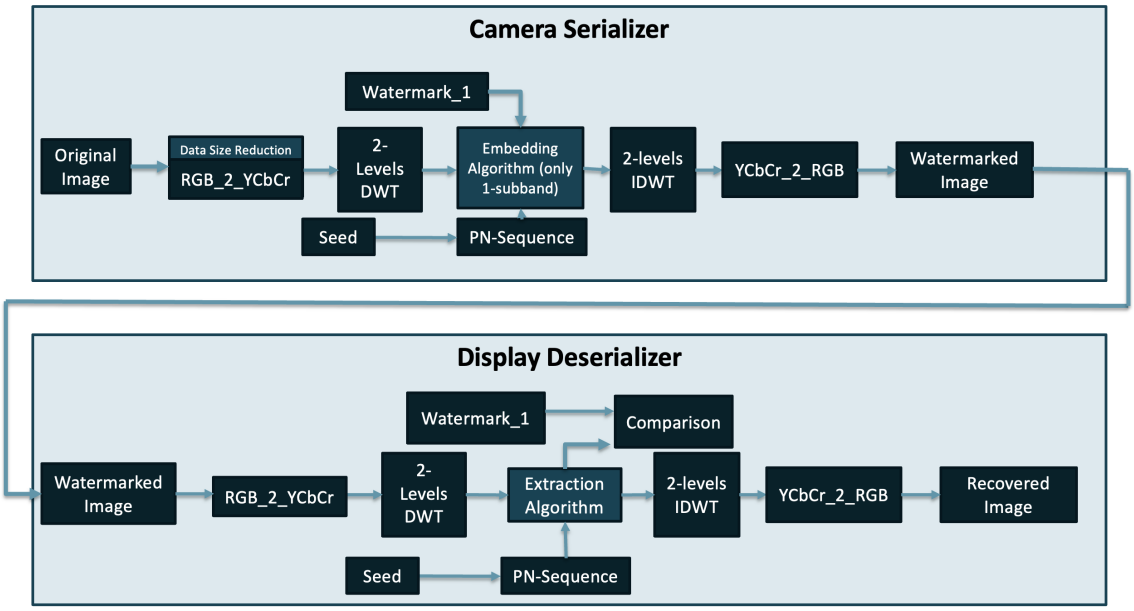


Figure 4.1. Base Effective Algorithm

- *Embedding:* The watermark is embedded in the LL subband of the image or frame's DWT. It is one of the most elementary method, whereby one embeds the watermark into the low-frequency subband for the purpose of providing stability.
- *Extraction:* DWT is applied on watermarked image. Watermark is retrieved from LL subband. Retrieved watermark is compared to original watermarked saved in display deserializer with the help of NCC parameter.

4.2.2. N-Sequenced Single Subband Watermarking (Shown in Figure 4.2)

N-Sequenced Single Subband Watermarking enhances Integrity since it can detect dropped frames, it is a type of tamper detection, and thus the video remains intact.

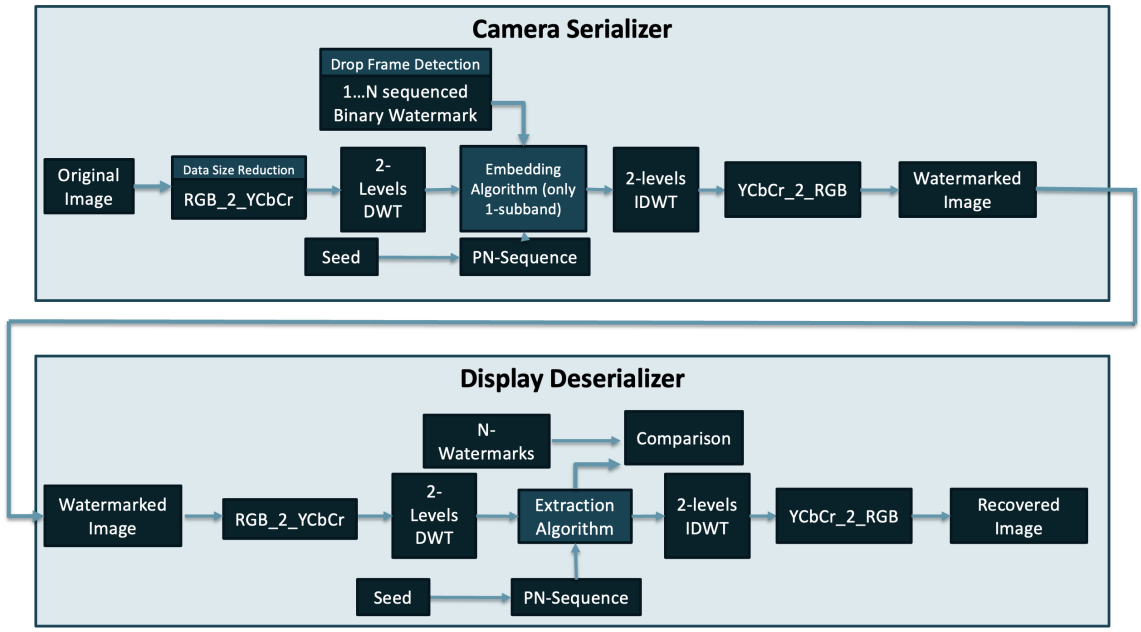


Figure 4.2. Sequenced Watermarking

- *Embedding:* N watermarks are embedded in a series of N consecutive frames, with each watermark being embedded in the LL subband. Once N frames are watermarked, the process is iterated for the next frames.
- *Extraction:* The watermarks are subsequently extracted from the frame sequence, and NCC is used to calculate how well each of the extracted watermarks approximates its original counterparts that are stored in the display deserializer.

4.2.3. HMAC Embedded Single Subband Watermarking (Shown in Figure 4.3)

HMAC Embedded Single Subband Watermarking supports Authentication and Non-repudiation, as HMAC ensures that verification and extraction of the watermark can only be done by legitimate parties.

- *Embedding:* An HMAC (Hash-based Message Authentication Code) is calculated using the watermark as a seed and implanted in the LL subband, predominantly for the authentication of the watermark rather than its integrity.

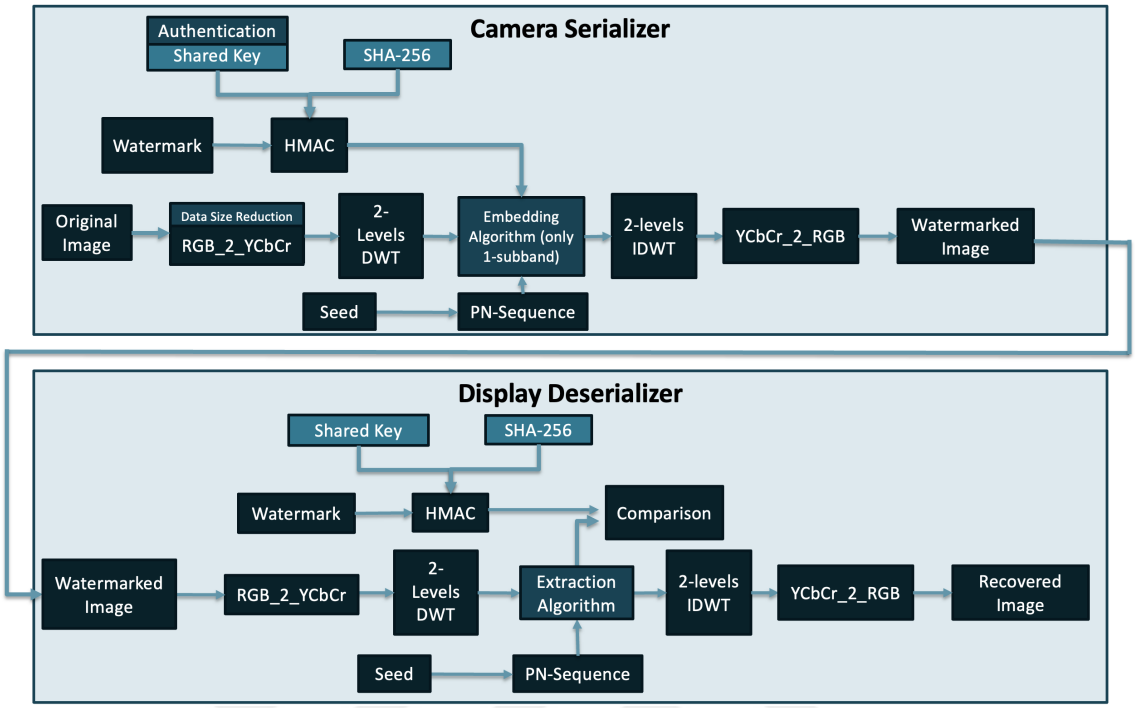


Figure 4.3. HMAC Embedded Watermarking

- *Extraction:* HMAC is extracted by the deserializer, and HMAC is re-computed from the original watermark that is stored in the display deserializer in order to maintain the integrity of the watermark. Mismatch between extracted and generated HMAC shows possible tampering.

4.2.4. Multi-Subband Watermarking (Shown in Figure 4.4)

Multi-Subband Watermarking is a trade-off between Integrity and Availability, since distributing the watermark across multiple subbands improves resistance to fault scenarios and attacks without affecting detectability.

- *Embedding:* The watermark is embedded in three subbands: LL, LH, HL. This gives redundancy in different frequencies, increasing robustness against certain types of attacks.
- *Extraction:* Watermarks are extracted from all subbands, and their similarity to the original watermark is evaluated by NCC. The maximum of the NCC values is to be selected in order to ensure robustness averaging out the effect of noise

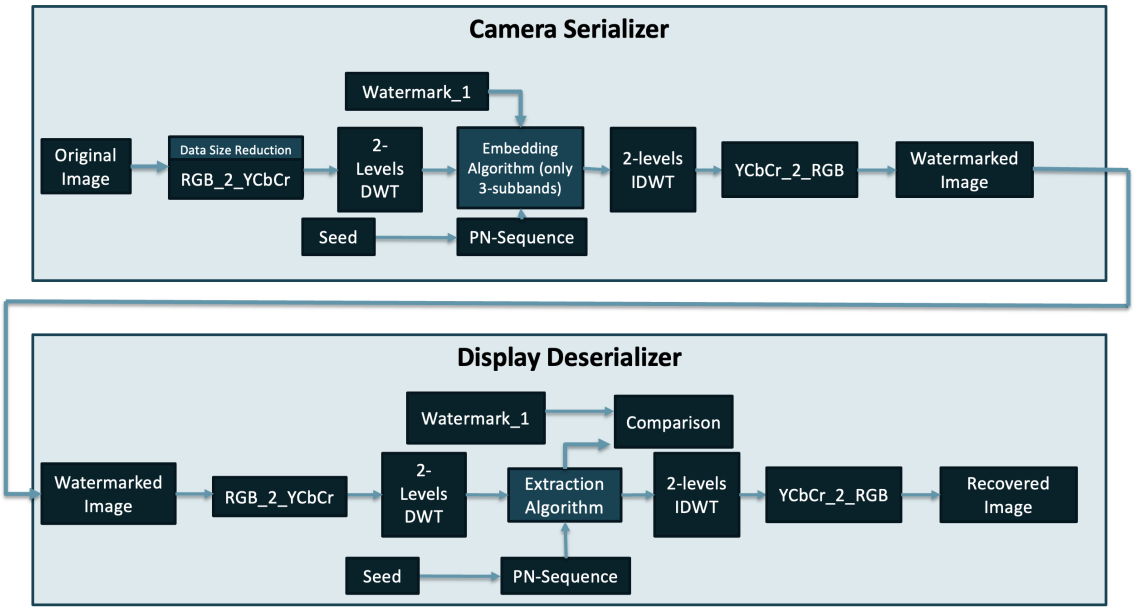


Figure 4.4. Multi-Subband Watermarking

or degradation in individual subbands, which tends to reduce both false positives and false negatives.

5. EXPERIMENTS AND RESULTS

In the examination of the performance of the proposed watermarking algorithms, a number of performance metrics are crucial for the understanding of the efficiency and robustness of the system. This chapter describes the metrics that are used in measuring the quality of the watermarked media and also the accuracy of the watermark detection. They include image quality metrics that are broadly used, correlation metrics for watermark detection accuracy. After the quantitative analysis we also mention the security features and robustness against attacks.

5.1. Performance Metrics

Peak Signal to Noise Ratio (PSNR) gives the quality of the watermarked media compared to its original, non-watermarked version. PSNR is obtained from Mean Squared Error (MSE), which is the average squared difference between the original and watermarked media. The larger the value of PSNR, the higher the imperceptibility of the watermarked multimedia. Conversely, a high value of PSNR means that the watermark is inserted in a way that will not have any noticeable effect on the perceptual quality of the media and hence the watermark will not be easily perceptible by human eyes. This property of imperceptibility is crucial for protecting the user experience and to ensure that the embedding will not reduce the viewing quality. Due to the attainment of high PSNR values, the watermarking method ensures that the embedded watermark has minimal perceptual impact but remains detectable and resilient to various fault scenarios and attacks. PSNR can be calculated as below,

$$\text{PSNR} = 10 \cdot \log_{10} \left(\frac{\text{MAX}^2}{\text{MSE}} \right). \tag{5.1}$$

MSE measures the amount of distortion introduced by the watermarking process. A lower MSE value indicates minimal distortion, helping maintain the integrity of the original media.

MSE is calculated as below,

$$\text{MSE} = \frac{1}{mn}\sum_{i=0}^{m-1}\sum_{j=0}^{n-1}[I(i,j) - K(i,j)]^2. \tag{5.2}$$

Normalized Correlation Coefficient (NCC) expresses the resemblance of the original watermark to the one which is extracted. The high NCC value, close to 1, indicates successful extraction, while the low NCC value near zero indicates detection failure or false detection. This measure is handy for the presence of the watermark and also suitable for finding its robustness. Robustness is defined as the ability of the watermark to be resistant against various forms of attacks and fault scenarios or manipulations, including compression, noise, cropping, and other forms of distortion. In these instances, a resilient watermarking system will guarantee that despite alterations occurring in the media, the watermark is still in place and can be detected. By maintaining the NCC values high for various test conditions, the robustness and reliability of the watermarking algorithm are guaranteed to allow extraction and correct identification of the embedded watermarks even in tampering or media degradation. NCC is calculated as such,

$$\text{NCC} = \frac{\sum_i \sum_j (WM_o[i,j] - \overline{WM_o})(WM_r[i,j] - \overline{WM_r})}{\sqrt{\sum_i \sum_j (WM_o[i,j] - \overline{WM_o})^2}\sqrt{\sum_i \sum_j (WM_r[i,j] - \overline{WM_r})^2}}. \tag{5.3}$$

In the MATLAB implementation, the calculation for NCC and PSNR can be seen under Appendix A.1.3.

Computational complexity is the hardness of a problem in terms of the number of operations needed, as opposed to real time, which also involves hardware and software considerations. It is usually spoken of in terms of deterministic Turing machines—ideal devices that perform one step at a time following fixed rules—and aids in the categorization of problems by theoretical work performed. [24] For watermarking in automotive systems, computational complexity can be an important factor. It specifies the processing time and power the algorithm requires—particularly crucial in real-time systems where latency can compromise functional safety [25].

Whereas theoretical analysis generally resorts to Big-O notation for the expression of algorithm scalability, in this work we take an empirical runtime analysis more suited for practical assessment [26]. To determine computational cost:

- We record the total runtime of the complete watermark embedding and extraction process with MATLAB's `tic` and `toc` functions [27].
- This is repeated for ten cycles, and we notice:
  - The average runtime as a measure of average performance.
  - The standard deviation to calculate the consistency or variability in runtime across those 10 trials.

This mirrors real-world feasibility, whereby low runtime variance implies reliability for predictable system behavior—vital in automotive applications [28].

These metrics are gathered in the primary implementation loop as can also be seen in Appendix A.1.3), and they allow for the computation of computational efficiency along with robustness (NCC) and imperceptibility (PSNR).

5.2. Introduction to Fault Scenarios and Attacks

In digital watermarking, the capacity to withstand fault scenarios and attacks in various forms and shapes is a critical requirement that determines the feasibility of both embedding and extraction. The algorithms discussed in Section 4 are designed to address these challenges. Fault scenarios and attacks in watermarking systems can either be unintentional (i.e., compression or addition of noise during transmission of videos) or intentional (i.e., trying to remove or modify the watermark). In that context, the robustness of the algorithms to the selected attacks is tested. These fault conditions or attacks can originate from numerous sources in the automotive video transmission chain. For instance, in Camera Monitoring Systems with SerDes links such as GMSL, video signals can travel through 15-meter-long coaxial or twisted pair cables. At these lengths, electromagnetic interference, insufficient shielding, or connector degradation may introduce transmission noise or cause compression-related quality loss. Also, ad-

versarial attacks can occur in more unexpected ways—such as when a vehicle is brought to a service center, and an untrusted mechanic inserts an adversarial external device into the internal video bus, attempting to manipulate or spoof the video stream. In autonomous vehicles, this type of tampering would lead to a spoofed visual environment, causing the system to make incorrect driving choices. These real-life scenarios demonstrate that both unintentional errors and malicious attacks must be considered in assessing the strength and robustness of watermarking algorithms.

This section evaluates the robustness of the proposed watermarking algorithms against common fault scenarios and deliberate attacks that may occur in automotive video transmission systems. The selected test scenarios include both unintentional degradations (compression artifacts, transmission noise) and intentional attacks (cropping, pixel replacement) that represent realistic threats in CMS. The following list details the implementation and characteristics of each fault and attack scenario.

- *Compression:* Compression is usually applied in video pipelines of cars before presentation or transmission, specifically when bandwidth or latency optimization is limited (e.g., for SerDes-based CMS structures). Lossy compression schemes like JPEG or H.264 discard high-frequency components—precisely where watermarks are usually embedded. This leads to watermark fidelity degradation that affects its perceptibility and robustness.

  In car video processing and on-board networking, testing the robustness of image processing or watermarking algorithms commonly involves simulation of real-world degradations. One of the most widely used methods for accomplishing this is JPEG compression, which creates perceptually important artifacts like blocking, blurring, and color banding because of its 8×8 block-based discrete cosine transform (DCT) and lossy quantization [29]. JPEG degradation can be introduced in MATLAB by invoking the `imwrite` function with a `Quality` parameter, enabling controlled introduction of compression artifacts. It is immensely useful in investigating the behavior of algorithms under bandwidth-limited scenarios or low-fidelity display conditions. During the simulations Quality will be set as 10. But JPEG does not describe the compression type utilized in standard high-

(a) Q = 10 (b) Q = 50 (c) Q = 100

Figure 5.1. Image under different Quality values

speed vehicle video links. In contemporary systems like GMSL, FPD-Link III, or MIPI-DSI/CSI interfaces, the de facto standard for real-time visually lossless compression is the VESA Display Stream Compression (DSC) protocol [30], [31]. DSC is quite different from JPEG: line-based, predictive, and ultra-low latency optimized with ratios typically between 2:1 to 3:1. It provides very little perceptual degradation while lowering transmission bandwidth by a considerable amount.

While comprehensive support for DSC is not native to MATLAB, approximations can be simulated through external C-based reference models from VESA or by simulating visually lossless behavior using high-quality JPEG 2000 or lightweight quantization schemes.

Thus, though JPEG-based simulation is adequate for preliminary testing and algorithmic stress testing, it is not necessary to simulate the real-world behavior of automotive-grade video pipelines based on DSC. For simulating embedded video compression and transmission more precisely, the utilization of DSC reference encoders or modeling line-based entropy coding is advisable. Figure 5.1 shows images under different quality values set in Matlab. Specifically, Figure 5.1 (a) shows the image with quality set to 10, Figure 5.1 (b) shows the image with quality set to 50, and Figure 5.1 (c) shows the image with quality set to 100.

- *Noise:* Noise refers to unwanted variations in a signal that is being transmitted, due to physical origins such as electrical interference, optical deviation, or electromagnetic noise. Due to physical limitations, it is not possible to construct a completely noise-free channel in real systems [32].

One method in which a statistical representation of noise is achievable is via a probability density function (PDF). Consider a collection of many noise samples and represent them as a histogram. As the sample size tends to infinity and the bin size goes to infinitesimally small, such a histogram tends to a continuous function known as a PDF [32].

Let X be a continuous random variable representing the noise. The PDF of X, denoted by $f_X(x)$, expresses the probability that X lies within a small interval $[x, x + dx]$, as,

$$P(x \leq X \leq x + dx) = f_X(x)\, dx. \tag{5.4}$$

The expected value (mean) μ_X of the random variable is computed using the PDF as follows,

$$\mu_X = \mathbb{E}[X] = \int_{-\infty}^{\infty} x\, f_X(x)\, dx. \tag{5.5}$$

Among the vast array of noise models, the Gaussian (normal) distribution holds particular significance in communication systems. Its significance is due to the Central Limit Theorem, which affirms that the sum of numerous independent random variables will approach a Gaussian distribution regardless of whether the individual variables are normally distributed or not [32].

The PDF of a Gaussian distribution is defined as,

$$f_X(x) = \frac{1}{\sqrt{2\pi\sigma^2}} \exp\left(-\frac{(x-\mu)^2}{2\sigma^2}\right). \tag{5.6}$$

Here, μ denotes the mean, and σ^2 is the variance. This formulation shows that values near the mean are more likely to occur, while values further away have exponentially lower probability [32].

White Gaussian noise is a random process having a flat power spectral density across all frequencies ("white") and amplitudes that have a normal distribution, also meaning μ equal to 0. In image and video processing, it is used to model independent and identically distributed (i.i.d.) additive noise to each pixel, e.g., to approximate channel transmission errors or sensor defects [32].

Noise can enter the system under long-distance in-vehicle transmission, particularly over up to 15-meter GMSL or FPD-Link cables. Interference sources like electromagnetic interference (EMI), harsh automotive conditions, or degradation of cable shielding can introduce Gaussian or impulse noise. Without any malicious intent, such natural types of disturbances test the watermark robustness in real driving conditions.

For the usage of Gaussian noise in the algorithms within this framework is the function below;

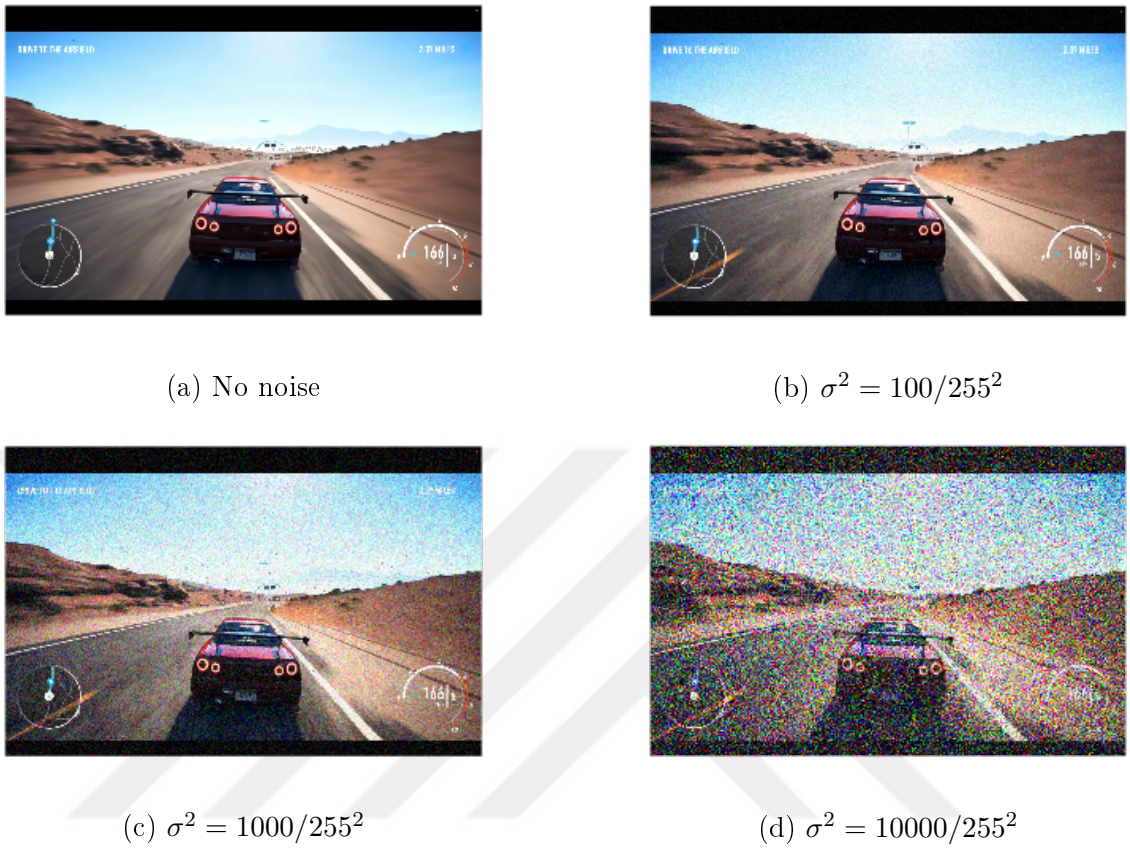
```
J = imnoise(I,"gaussian",m,variance)
```

The variance value that is provided in Matlab is for images where pixel values are between 0 and 1. However, most color images imported using `imread` are of the type uint8 and have a range of 0 to 255. If one gives a pixel that has such an image to `imnoise`, MATLAB will implicitly convert it to a double data type with values scaled to the interval [0,1] before it adds the noise. Therefore the noise variance must normalized to the [0,1] interval via dividing by 255^2 (since variance is proportional to the square of the intensity). This ensures that the added noise is within the normal range of normalized intensity.

Figure 5.2 shows images under different σ^2 values set in Matlab. Specifically, Figure 5.2 (a) shows the image with no noise, Figure 5.2 (b) shows the image with $\sigma^2 = 100/255^2$, Figure 5.2 (c) shows the image with $\sigma^2 = 1000/255^2$, and Figure 5.2 (d) shows the image with $\sigma^2 = 10000/255^2$.

- *Cropping:* Cropping may be either intentional (using a manipulated display device or adversarial preprocessing) or unintentional due to a hardware glitch. If only a portion of the frame is being shown or processed, and the watermark has not been redundantly embedded over the entire frame, recovery of the watermark becomes much tougher.

  Cropping attacks are the most frequent kind of deliberate distortion to digital images and videos in an attempt to establish the strength of watermarking algorithms. Cropping will be simulated here with the help of MATLAB by substituting a defined area of the image with uniform pixel values. The command `frame(1:256, 1:256, :) = 255;` fills the upper-left 256×256 part of the image with white and `frame(1:256, 1:256, :) = 0;` with black in all three color



(a) No noise

(b) $\sigma^2 = 100/255^2$

(c) $\sigma^2 = 1000/255^2$

(d) $\sigma^2 = 10000/255^2$

Figure 5.2. Image under different noise variances

channels of the RGB representation. It is a local distortion simulation that obscures all visual content in a spatially confined block. In terms of watermarking, this is a spatial-domain cropping attack, especially when the watermark is embedded within the cropped area. The major impact of such an attack is to annihilate or incapacitate the embedded watermark bits within that area, which can notably degrade extraction quality or lead to overall watermark detection failure. These kinds of attack are normally employed to evaluate spatial and frequency domain watermarking methods for robustness to localized data loss [27], [33], and to simulate effects like occlusion, censoring, or transmission damage to the frames. The degree to which a watermark is resistant to this kind of attack relies on the embedding domain (i.e., spatial vs. transform), the extent of the watermark spread, and on any redundancy mechanism that is used. Figure 5.3 shows cropping with the pixels set as 0 and 255 in Matlab. Specifically, Figure 5.3 (a) shows the image with the pixel values set to 0, and Figure 5.3 (b) shows the image with the pixel

(a) Value set as 0 (b) Value set as 255

Figure 5.3. Image under cropping

values set to 255.

- *Pixel Replacement Attack:* One extreme form of deliberate tampering can be exemplified by replacing the entire video frame with black pixels, done in MATLAB using `frameRGB(:,:,:) = 0;`. This simulates a scenario where an attacker—e.g., malicious technician or compromised ECU—injects artificial video frames into the transmission channel. The pixel replacement can be uniform across the frame, fully suppressing visual content, or selective to target regions of watermark embedding. Replacement of the whole frame signals severe adversarial conditions like deepfake injection, video spoofing, or adversarial perturbations that are especially crafted to evade AI-based perception systems. Such attacks are critical in automotive and intelligent transportation systems as they directly undermine the watermark's function in integrity verification and authenticity assurance [28], [34], [35]. The watermark signal will be completely annihilated or unrecoverable when the pixel values are fully zeroed, and it presents itself as a benchmark to check the robustness of watermark extraction algorithms against adversarial, real-time manipulation conditions.

.

5.3. Experimental Results

In this section, experimental results will be presented to evaluate the performance of the four presented algorithms on these metrics. PSNR values are considered for media quality and NCC for watermark detectability accuracy. Each algorithm's trade-offs between robustness and performance will be discussed in detail in the next chapter. Simulations are performed using MATLAB from 2 different videos, Video 1 being less active and has a darker color tone and Video 2 has contrary features. Each video consists of 120 frames (2 seconds). As a baseline, PSNR value for non-watermarked Video 1 is 48.740 ± 0.011 and for Video 2 is 48.058 ± 0.089 over 120 frames.

Simulations are conducted on 8-core, 2.4 GHz, 8 GB RAM and 256 GB SSD personal computer. Each algorithm is run 10 times by resetting the random number generator with a different seed. For each result, the average value as well as the standard deviations are given. Because watermark strength is affected by Gaussian noise and the random nature of the pseudo-random spreading sequences, each algorithm is executed ten times. For each experiment we monitor the frame-by-frame PSNR and NCC values; finally we report their grand-mean and standard deviation across the ten experiments. This Monte Carlo sampling methodology provides an unbiased empirical estimate of the mean perceptual quality and extraction reliability under random channel conditions [36, 37]. For the measurement of the run-time duration, in order to minimize the effect of outlier values and runtime variability, a trimmed mean approach is utilized. The maximum and minimum time/duration measurements are omitted, and the average and standard deviation calculated on the remaining 8 samples. This is equivalent to a 10% trimmed mean from both tails ($\alpha = 0.1$), a standard procedure in robust statistics for improving the reliability of central tendency estimates in small-sample performance testing [38, 39].

Table 5.1 shows the PSNR and NCC values for different algorithms for Video 1 and Video 2. The PSNR of Video 1 is around 0.21% higher than Video 2 in the Base Effective Algorithm. For all methods, the maximum PSNR for both videos is obtained by the Base Effective Algorithm, indicating the watermark has higher imperceptibility.

Table 5.1. Comparison of PSNR and NCC for different watermarking methods across two videos.

Method	*Video 1 PSNR (dB)*	*Video 2 PSNR (dB)*	*Video 1 NCC*	*Video 2 NCC*
Base Effective	37.073 ± 0.011	36.999 ± 0.011	0.689 ± 0.018	0.684 ± 0.018
Sequenced	36.841 ± 0.011	36.772 ± 0.010	0.654 ± 0.020	0.653 ± 0.019
Multi-Subband	32.532 ± 0.007	32.486 ± 0.006	0.700 ± 0.013	0.700 ± 0.013
HMAC Embedded	28.306 ± 0.012	28.490 ± 0.012	0.641 ± 0.006	0.646 ± 0.006

But it lacks essential cybersecurity features—it doesn't have drop-frame detection, authentication, non-repudiation, or availability features which are essential in automotive cybersecurity. On a runtime performance point of view, the Sequenced Watermarking is the fastest executing algorithm, and consequently the most appropriate for use in soft real-time systems (see Table 5.2). The HMAC Embedded is the most time-consuming as it entails additional cryptographic computation overhead. The Multi-Subband solution is also the slowest executing since there is the need to embed the watermark in more than one frequency subband. The runtime standard deviations of PSNR and NCC across the 10 Monte Carlo runs are generally low for all methods as shown in Table 5.2, indicating consistency and reliability in image quality and watermarking robustness. However, the standard deviations of runtime are comparatively more different, with the HMAC method having the highest variability, which means that it may not guarantee consistent timing performance—a critical requirement for real-time automotive systems. Lastly, keep in mind that all performance figures were measured on a limited hardware configuration (8-core CPU, 8GB RAM), which could restrict generalizability. Thus, particularly for runtime measures, these figures might not reflect true performance on optimized embedded platforms and are to be used with some reservation. These results only give comparative perspective for the tested algorithms.

Figure 5.4 shows the PSNR values for Video 2 over the 120 frames and Figure 5.5 shows the NCC values. For Base Effective algorithm it shows approximately 22.9% drop on the PSNR values. Furthermore, as it can also be seen in Table 4.1, the

Table 5.2. Runtime Performance of Each Method (Average ± Std Deviation)

Method	*Video 1 (s)*	*Video 2 (s)*
Base Effective	123.372 ± 5.890	130.776 ± 5.689
Sequenced	84.934 ± 3.656	68.757 ± 5.587
Multi-Subband	165.208 ± 2.925	167.326 ± 3.553
HMAC Embedded	285.051 ± 7.847	317.155 ± 4.665

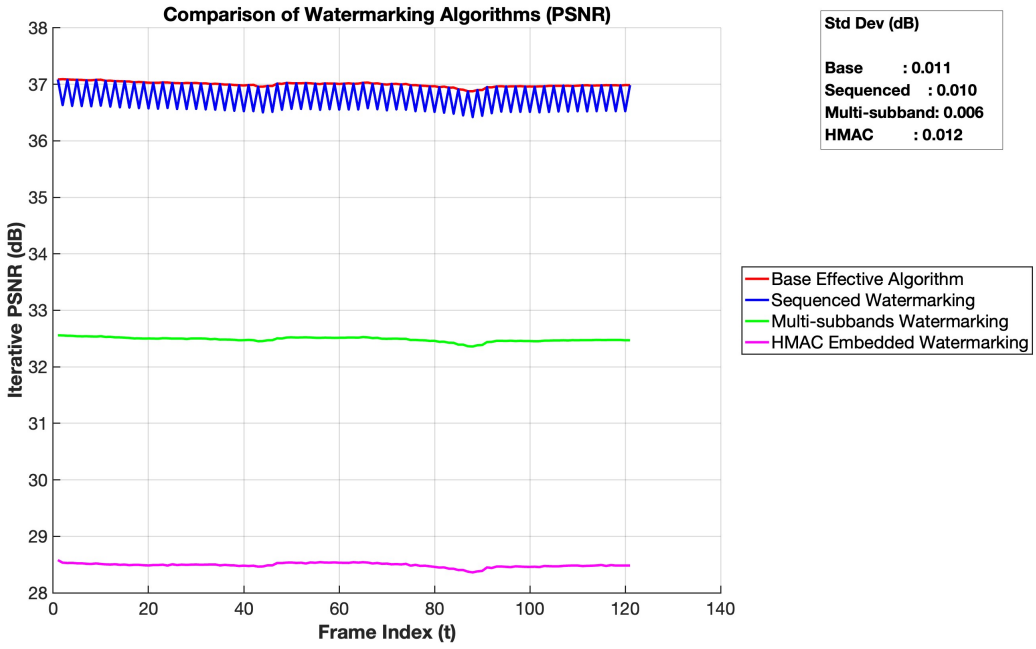


Figure 5.4. PSNR Comparison Graph

Multi-Subband Watermarking algorithm has about 10% lower PSNR value in average compared to the Base Effective Algorithm having Single Subband Watermarking and with the smallest, the HMAC Embedded Single Subband Watermarking has about 23% lower PSNR value in average. In addition, a Zigzag pattern can be seen for Sequenced watermarking in Figure 5.5. It might be due to different watermark values, where some watermark patterns are more correlated in a frame than others. Sliding window effect could be the second possible reason. In a sliding window (say, N=4), frames are treated in sets, and this induces periodic NCC variations. Buffering delay could also desynchronize frames with their anticipated watermark, reducing correlation.

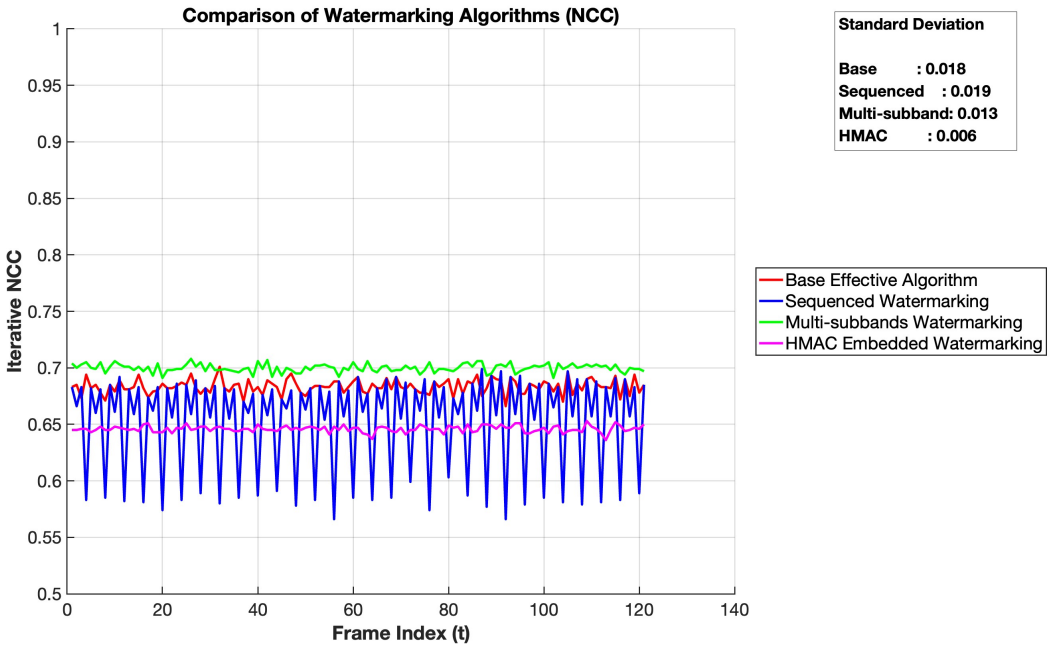


Figure 5.5. NCC Comparison Graph

Table 5.3. PSNR for Base Effective Method under GN (σ^2=100) + Attack Scenarios

Scenario	*Video 1 (dB) Mean ± Stdev*	*Video 2 (dB) Mean ± Stdev*
GN + Cropping	14.109 ± 0.001	15.269 ± 0.001
GN + Compression	26.374 ± 0.003	26.832 ± 0.006
GN + Blackout	20.354 ± 0.001	19.028 ± 0.001

Overlapping or non-overlapping buffers can contain residues from the previous cycles, retaining the zigzag pattern.

The results in Table 5.3 shows the Average PSNR (dB) values for Video 1 and Video 2 for different fault scenarios & attack scenarios. Most significant take-aways for Base Effective method from Table 5.3, Table 5.4 and Table 5.5 regarding the two videos are:

- For the highest PSNR valued Base Effective Algorithm, visual quality is degraded by approximately 22.9%.

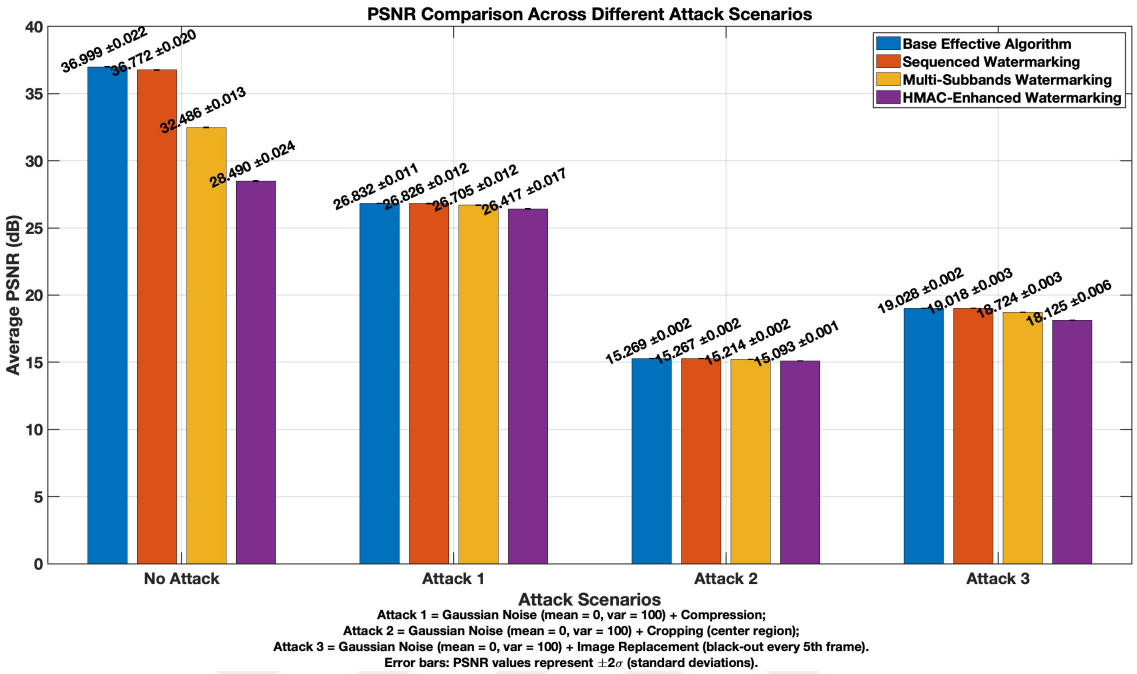


Figure 5.6. Average PSNR Values under different fault scenarios & attacks

Table 5.4. NCC for Base Effective Method under GN (σ^2=100) + Attack Scenarios

Scenario	*Video 1 Mean ± Stdev*	*Video 2 Mean ± Stdev*
GN + Cropping	0.684 ± 0.017	0.685 ± 0.018
GN + Compression	0.684 ± 0.018	0.686 ± 0.018
GN + Blackout	0.685 ± 0.017	0.685 ± 0.018

- Video 1 consistently has a greater PSNR value compared to Video 2 for all attacks and methods.
- The largest PSNR gap occurs in the "Image Replacement" attack, where Video 1 is roughly 14.5% better than Video 2.
- Under the Gaussian Noise & Cropping configuration, the variance remains moderate, some 2.2–2.3%.
- When using Gaussian Noise & Compression, the variance is moderate at 7.5–8.1%.
- These results suggest that Video 1 is less likely to suffer from visual degradation, perhaps due to diverging content dynamics, resolution, and inherent compression.

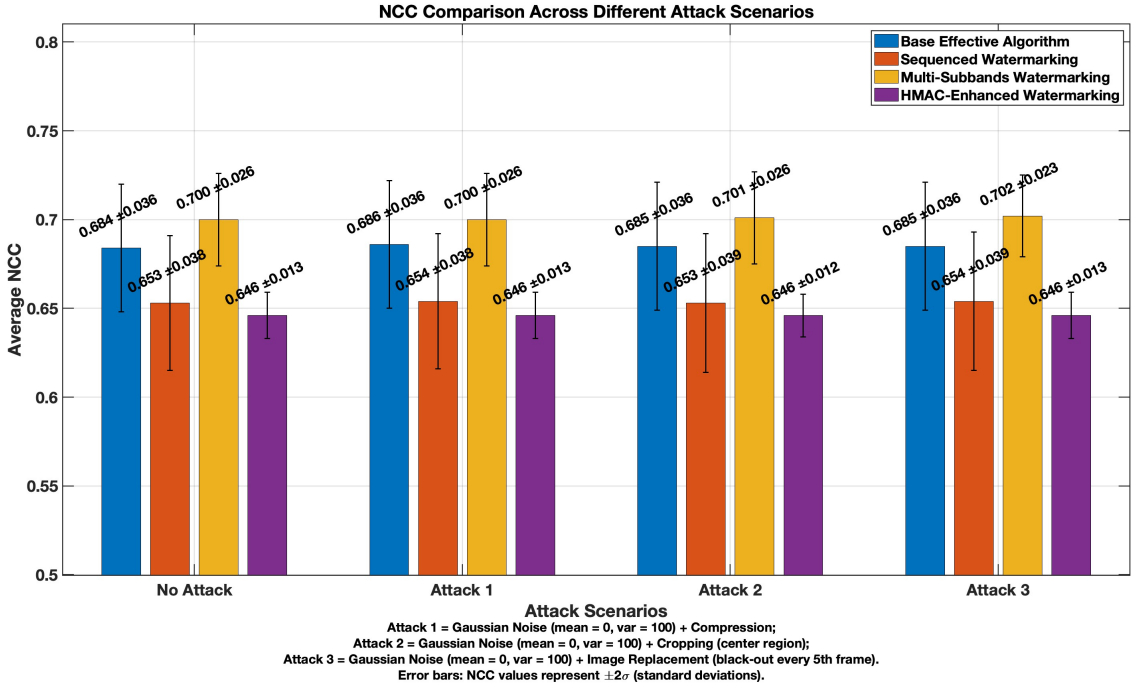


Figure 5.7. Average NCC Values under different fault scenarios & attacks

Table 5.5. Runtime for Base Effective Method under GN (σ^2=100) + Attack Scenarios

Scenario	*Video 1 (s) Mean ± Stdev*	*Video 2 (s) Mean ± Stdev*
GN + Cropping	320.69 ± 108.50	214.67 ± 78.76
GN + Compression	208.75 ± 55.12	253.12 ± 125.29
GN + Blackout	300.62 ± 76.21	179.26 ± 20.11

- Computationally, the GN + Cropping and GN + Blackout scenarios are much more time-consuming, especially for Video 1.
- Some runtime standard deviations are extremely large (e.g., > 100 seconds), reflecting variability most likely caused by system resource limitations and MATLAB's execution overhead on the test platform. These runtime values should be interpreted with caution and just consider comparatively between cases.

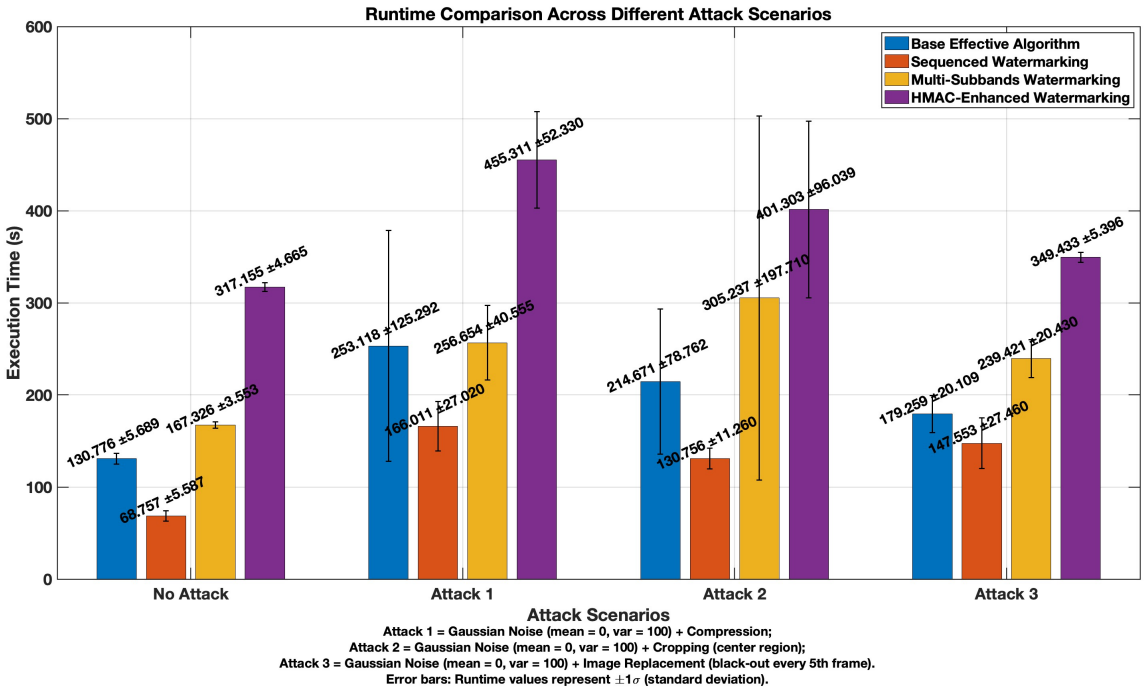


Figure 5.8. Average Runtime Values under different fault scenarios & attacks

The results in Figure 5.7, Figure 5.6 and Figure 5.8 show the PSNR, NCC and Runtime values for different watermarking algorithms under a variety of fault and attack scenarios for Video 2.

They represent the performance of the watermarking algorithm. We analyze the detection probability and NCC for different fault scenarios & attack conditions and compare them with the original video when there is no watermarking. When there is no watermark, the detection probability and NCC are both zero, indicating no detectable watermark. When watermarking is applied, terms such as NCC and probability of detection can be considered. These are the best that can be achieved when compared with other sets of watermarking scenarios.

It can be concluded from the results that,

- Under mild Gaussian noise (σ^2=100), all the algorithms maintain high watermark correlation and good visual quality.

Table 5.6. PSNR (dB) ± Std for Video 2 under GN Variance 100 and 1000

Method	*Compression*	*Cropping*	*Blackout*
	GN Variance (σ^2) = 100		
Base Eff.	26.83 ± 0.006	15.27 ± 0.001	19.03 ± 0.001
Sequenced	26.83 ± 0.006	15.27 ± 0.001	19.02 ± 0.001
Multi Sub.	26.71 ± 0.006	15.21 ± 0.001	18.72 ± 0.001
HMAC	26.42 ± 0.008	15.09 ± 0.001	18.13 ± 0.003
	GN Variance (σ^2) = 1000		
Base Eff.	21.57 ± 0.009	12.07 ± 0.001	11.47 ± 0.001
Sequenced	21.56 ± 0.009	12.07 ± 0.001	11.47 ± 0.001
Multi Sub.	21.48 ± 0.009	12.04 ± 0.001	11.42 ± 0.001
HMAC	21.30 ± 0.009	11.98 ± 0.001	11.34 ± 0.001

- Base Effective Algorithm yields the best composite performance (best PSNR and NCC) when the fault or attack is very minor.
- More powerful methods (like HMAC-Enhanced Watermarking) partly compromise visual quality in these situations without offering much robustness benefits at this noise level.
- Runtime tends to get less unreliable and have more deviations for the attack scenarios.

After comparing algorithm performances under light Gaussian noise, the following tables extend this analysis to environments where there are extreme noise situations and illustrate key differences in terms of visual quality and watermark resilience.

For Table 5.6, Table 5.7, and Table 5.8, the comparison for Gaussian Noise (σ^2) values of 100 and 1000 for Video 2 is shown.

In watermarking, trade-offs between visual fidelity and robustness are expressed in terms of performance metrics such as PSNR and NCC. The combined analysis based on different Gaussian noise levels is provided as follows:

Table 5.7. NCC ± Std for Video 2 under GN Variance 100 and 1000

Method	*Compression*	*Cropping*	*Blackout*
	GN Variance (σ^2) = 100		
Base Eff.	0.686 ± 0.018	0.685 ± 0.018	0.685 ± 0.018
Sequenced	0.654 ± 0.019	0.653 ± 0.019	0.654 ± 0.019
Multi Sub.	0.700 ± 0.013	0.701 ± 0.013	0.702 ± 0.011
HMAC	0.646 ± 0.006	0.646 ± 0.006	0.646 ± 0.006
	GN Variance (σ^2) = 1000		
Base Eff.	0.686 ± 0.018	0.686 ± 0.017	0.686 ± 0.017
Sequenced	0.655 ± 0.020	0.654 ± 0.019	0.654 ± 0.019
Multi Sub.	0.700 ± 0.012	0.701 ± 0.012	0.702 ± 0.010
HMAC	0.646 ± 0.006	0.646 ± 0.006	0.646 ± 0.006

Table 5.8. Runtime ± Std for Video 2 under GN Variance 100 and 1000

Method	*Compression*	*Cropping*	*Blackout*
	GN Variance (σ^2) = 100		
Base Effective	253.12 ± 125.29	214.67 ± 78.76	179.26 ± 20.11
Sequenced	166.01 ± 27.02	130.76 ± 11.26	147.55 ± 27.46
Multi-Subbands	256.65 ± 40.56	305.24 ± 197.71	239.42 ± 20.43
HMAC Embedded	455.31 ± 52.33	401.30 ± 96.04	349.43 ± 5.40
	GN Variance (σ^2) = 1000		
Base Effective	287.85 ± 93.05	145.12 ± 8.39	205.55 ± 24.49
Sequenced	257.21 ± 36.02	114.76 ± 0.64	124.35 ± 6.99
Multi-Subbands	256.73 ± 12.29	231.38 ± 5.60	188.26 ± 3.93
HMAC Embedded	379.44 ± 2.06	350.03 ± 1.85	360.05 ± 5.71

- *No Fault & Attack.* The highest imperceptibility is achieved by the *Base Effective* algorithm, yielding 37.07 dB on *Video 1* and 36.99 dB on *Video 2*. Across all four schemes, NCC values remain within 0.641−0.700, confirming that each watermark can be detected correctly.
- *Moderate Gaussian Noise ($\sigma^2 = 100$).* Under a light-to-moderate noise level the visual quality degrades to around 20 dB: e.g. GN+Compression delivers ≈ 26.8 dB. NCC stays relatively high (0.646−0.702). *Base Effective* retains the best PSNR, whereas *Multi-Subband* gives the highest NCC, verifying its robustness at only a minor visual-quality degradation. Execution-time overheads are modest (e.g. 114−320 s), with Sequenced the quickest and HMAC the slowest.

- *Severe Gaussian Noise ($\sigma^2 = 1000$) + Compression.* PSNR drops further to ~ 21.5 dB for all schemes. Although *Base Effective* performs slightly better in terms of PSNR (21.57 dB), *Multi-Subband* secures the best NCC (0.700 ± 0.012), demonstrating better watermark detectability under heavy compression. Average runtimes roughly double compared with the moderate case, with standard deviations indicating larger variability (e.g. HMAC: 379.4 ± 2.1 s).
- *Severe Gaussian Noise ($\sigma^2 = 1000$) + Cropping.* Spatial removal is highly destructive: PSNR decreases to $\approx$ 12.07 dB. Despite the poor visual quality, watermark recovery is still feasible— *Multi-Subband* achieves the highest NCC (0.701 ± 0.012), followed by *Base Effective* (0.686 ± 0.017). Sequenced remains the least expensive computationally (115 ± 0.6 s), whereas HMAC's cryptographic overhead prolongs execution (350 ± 1.9 s).
- *Severe Gaussian Noise ($\sigma^2 = 1000$) + Blackout (Image Replacement).* When portions of the frame are fully overwritten, PSNR settles near 11.4 dB. Yet *Multi-Subband* again preserves the strongest correlation (0.702 ± 0.010), indicating that its redundancy across sub-bands is effective even against destructive tampering. Runtime variability increases substantially for high-complexity methods (e.g. *Multi-Subband* 239 ± 20 s), highlighting the need for optimization on resource-limited ECUs.

Based on these observations, the final conclusions are:

- *Best Visual Quality:* For low noise and no fault or attack conditions, the Sequenced Watermarking and Base Effective Algorithm consistently give maximum PSNR with lowest perceptual degradation. For very high noise ($\sigma^2 = 1000$), PSNR differences among approaches become negligible, and visual quality significantly degraded for all. Watermarking causes a drop of more than 20%.
- *Maximum Detection Resilience:* Multi-Subband Watermarking consistently obtains the highest NCC values in all heavy fault scenarios or attack instances. It ensures greater watermark robustness and can be particularly beneficial for maintaining watermark detection rates under destructive fault scenarios and attacks.

- *Best Trade-off Strategy:* Sequenced Watermarking is employed to ensure high image quality and can ensure Drop Frame detection. For heavy fault scenarios or attack conditions, Multi-Subband Watermarking is useful due to its greater detection reliability.

Overall, the watermarking method should be decided based on the expected threat level and robustness under heavy fault scenarios or attack conditions. Please note that, if you need authentication as a functional requirement you have to apply HMAC or similar approach integrated to watermarking scheme which introduces additional computational overhead. The performance analysis shown in this thesis can be a comparative quantification for the performance metrics showing the tradeoff of using HMAC or some other technique to improve the security.

6. CONCLUSION AND FURTHER DIRECTIONS

In this thesis, a comprehensive evaluation of DWT-based video watermarking techniques was conducted in the context of their robustness against common video processing artifacts and cybersecurity-related attacks. The proposed algorithms were developed and evaluated under various conditions, including Gaussian noise, cropping, blackout, and compression distortions. The Base Effective Algorithm was the foundation, with three extended versions — Sequenced Watermarking, Multi-Subband Watermarking, and HMAC-Enhanced Watermarking — developed to offer differing aspects of watermark robustness and cybersecurity requirements.

To ensure the reliability of outcomes, the simulations were executed 10 times per scenario with different seeds. This allowed the PSNR (Peak Signal-to-Noise Ratio), NCC (Normalized Cross-Correlation), and runtime to be compared both in terms of average and standard deviations. Results indicated that while the Base Effective Algorithm consistently achieved the best visual quality (PSNR) in low or no-fault conditions, its robustness against attacks was limited. On the other hand, Multi-Subband Watermarking achieved the highest NCC values for most of the severe attack cases, particularly for heavy Gaussian noise combined with blackout or compression, reflecting its highest detection reliability.

Standard deviation results revealed interesting variability in performance across schemes. Schemes such as Multi-Subband Watermarking and HMAC-Enhanced Watermarking exhibited relatively steady NCC values but experienced greater variability in runtime, indicating higher computational complexity. Execution time analysis also revealed that more elaborate schemes such as HMAC, while providing enhanced security, require significantly more processing time, which would strain real-time viability in embedded systems.

It was also observed that the type of video content plays a significant role in the outcome. Videos with less motion and darker content preserved visual quality and watermark integrity more effectively under distortion. Videos with high color content and motion, however, experienced greater degradation, reducing both PSNR and NCC in case of a fault.

The results affirm that the proposed technique can be applied practically in real-time automotive video systems where the integrity and authenticity of video must be ensured under adverse operating conditions. Even though the visual quality is decreased, the proposed watermarking algorithms augment the data security principles of ISO/SAE 21434, which standardizes integrity, authentication, and confidentiality requirements in in-vehicle systems and communication.

This thesis lays a sound foundation for future enhancements. Further improvement can be achieved by employing adaptive watermarking schemes that vary watermark strength according to content and noise. Incorporation of cryptographic primitives in watermark generation and verification could further improve security guarantees. Hardware-in-the-loop (HIL) testing frameworks or real embedded systems would allow verification against realistic latency and resource constraints.

Overall, while the suggested methods achieve better performance in robustness, imperceptibility, and runtime in a simulation framework, further research and development are needed to drive the system further towards large-scale real-time and secure deployment in automotive and other mission-critical applications.

REFERENCES

1. Gutzmer, R., *Video Interface Technology*, Springer International Publishing AG Switzerland, Cham, Switzerland, 2016.

2. Analog Devices, "GMSL2 Hardware Design Guide", 2023, `\url{https://www.analog.com/media/en/technical-documentation/user-guides/gmsl2-channel-specification-user-guide.pdf}`, accessed on October 10, 2024.

3. International Organization for Standardization (ISO) and Society of Automotive Engineers (SAE), "Road Vehicles – Cybersecurity Engineering", ISO/SAE 21434:2021, 2021, accessed on November 19, 2024.

4. MathWorks, "DWT2", 2024, `\url{https://www.mathworks.com/help/wavelet/ref/dwt2.html}`, accessed on December 2, 2024.

5. Texas Instruments, "Texas Instruments DS90UB633A-Q1 FPD-Link III Serializer", 2024, `https://www.mouser.com.tr/new/texas-instruments/ti-ds90ub633a-q1-serializer/`, accessed on March 9, 2025.

6. Roemer, M., *APIX: High Speed Automotive Pixel Link*, Springer, Berlin, Heidelberg, 2012.

7. Terzis, A., *Automotive Mirror-Replacement by Camera Monitor Systems*, Springer International Publishing AG Switzerland, Cham, Switzerland, 2016.

8. Witt, K. and J. Bauer, "A Robust Method for Frozen Frame Detection in Safety Relevant Video Streams Based on Digital Watermarking", *Maxim Integrated*, pp. 1–6, 2023.

9. Axmann, B., F. Langner, K. Blankenbach, M. Vogelmann, A. Gaisberg, L. Strobel

and J. Bauer, "Advanced Methods for Safe Visualization on Automotive Displays", *Journal of the Society for Information Display*, Vol. 28, pp. 483–498, 2020.

10. Rahinj, K. S., "Literature Review on Watermarking Techniques of Relational Databases", *International Journal of Innovations in Engineering Research and Technology (IJIERT)*, Vol. 4, No. 8, 2017.

11. Melman, A. and O. Evsutin, "Methods for Countering Attacks on Image Watermarking Schemes: Overview", *Journal of Visual Communication and Image Representation*, Vol. 99, p. 104073, 2024.

12. Agbaje, M., A. Oludele and C. Ogbonna, "Applications of Digital Watermarking to Cyber Security (Cyber Watermarking)", *Proc. of InSITE 2015: Informing Science + IT Education Conferences*, pp. 1–11, Tampa/Florida, USA, 2015.

13. Wu, W., J. Dai, H. Huang, Q. Zhao, G. Zeng and R. Li, "A Digital Watermark Method for In-Vehicle Network Security Enhancement", *IEEE Transactions on Vehicular Technology*, Vol. 72, No. 7, pp. 8398–8408, 2023.

14. Bomey, N., "Backup Cameras Now Required in New Cars in the US", 2018, `\url{https://www.usatoday.com/story/money/cars/2018/05/02/backup-cameras/572079002/}`, accessed on December 20, 2023.

15. Zinner, H., "Automotive Ethernet and SerDes in Competition", *ATZ Electron Worldw*, Vol. 15, pp. 40–43, 2020.

16. Mehta, C., S. Lippincott and C. Lu, "Communication Protocols in Modern ADAS Architectures", 2023, `\url{https://www.ti.com/lit/SLYY219}`, accessed on November 21, 2024.

17. Ostrem, G., "GMSL: Gigabit Multimedia Serial Link - An Introduction", K. Blankenbach, Q. Yan and R. O'Brien (Editors), *Handbook of Visual Display Technology*, pp. 1–22, Springer, Berlin, Heidelberg, 2023.

18. Kim, S. and R. Shrestha, "Introduction to Automotive Cybersecurity", *Automotive Cyber Security*, pp. 1–13, Springer, Singapore, 2020.

19. Pranitha, G., "Comparison of MAC and Digital Signature", *International Journal of Science and Research*, Vol. 8, No. 2, pp. 50–52, 2019.

20. Gonzalez, R. C. and R. E. Woods, *Digital Image Processing*, Pearson, New Jersey, USA, 2018.

21. Mallat, S., *A Wavelet Tour of Signal Processing: The Sparse Way*, Academic Press, Boston, USA, Third Edition, 2008.

22. Ravichandran, D., R. Nimmatoori and M. G. Ahamad, "Mathematical Representations of 1D, 2D and 3D Wavelet Transform for Image Coding", *Institute for Research and Development India*, Vol. 5, pp. 2319–2526, 2023.

23. Cox, I., G. Doërr and T. Furon, "Watermarking is not Cryptography", Springer-Verlag (Editor), *Lecture Notes in Computer Science*, pp. 1–15, Jeju island, South Korea, 2006.

24. Yang, X.-S., *Mathematical Foundations for Algorithm Analysis*, Academic Press, Cambridge/Massachusetts, USA, 2020.

25. Kundur, D. and D. Hatzinakos, "Digital Watermarking Using Multiresolution Wavelet Decomposition", *International Conference on Acoustics, Speech, and Signal Processing*, pp. 2969 – 2972, IEEE, Ontario, Canada, 1999.

26. Katzenbeisser, S. and F. A. Petitcolas, *Information Hiding Techniques for Steganography and Digital Watermarking*, Artech House, Boston, USA, 2000.

27. Cox, I., M. Miller, J. Bloom, J. Fridrich and T. Kalker, *Digital Watermarking and Steganography*, Morgan Kaufmann, Massachusetts, USA, 2002.

28. Barni, M. and F. Bartolini, *Watermarking Systems Engineering: Enabling Digital*

Assets Security and Other Applications, CRC Press, Boca Raton/Florida, USA, 2001.

29. Wallace, G. K., "The Jpeg Still Picture Compression Standard", *IEEE Transactions on Consumer Electronics*, Vol. 38, No. 1, pp. 18–34, 1992.

30. VESA, "Display Stream Compression Specification, Version 1.2a", 2023, `\url{https://vesa.org/vesa-display-compression-codecs/}`, accessed on November 4, 2024.

31. VESA, "Visually Lossless Compression for Display Interfaces", 2020, `\url{https://vesa.org/vesa-standards/}`, accessed on December 7, 2024.

32. Massachusetts Institute of Technology, Department of Electrical Engineering and Computer Science, "6.02 DRAFT Lecture Notes, Fall 2010 — Lecture 4: Noise and Random Variables", 2010, `\url{https://web.mit.edu/6.02/www/f2010/handouts/lectures/L4-notes.pdf}`, accessed on April 3, 2025.

33. Voloshynovskiy, S., S. Pereira, T. Pun, J. Eggers and J. Su, "Attack Modeling: Towards a Second Generation Watermarking Benchmark", *Signal Processing*, Vol. 81, No. 6, pp. 1177–1214, 2001.

34. Pun, T., S. Voloshynovskiy, F. Deguillaume, S. Pereira and J. Eggers, *Information Security and Image Processing: Steganography, Digital Watermarking and Covert Communication*, Springer, Berlin, Heidelberg, 2004.

35. Wu, M. and B. Liu, "Current Developments in Watermarking Techniques for Digital Images: a Survey", *IEEE Signal Processing Magazine*, Vol. 33, No. 5, pp. 56–68, 2016.

36. Kalos, M. H. and P. A. Whitlock, *Monte Carlo Methods*, John Wiley & Sons, Darmstadt, Germany, Second Edition, 2008.

37. Robert, C. P. and G. Casella, *Monte Carlo Statistical Methods*, Springer, New

York, USA, Second Edition, 2004.

38. Wilcox, R. R., *Introduction to Robust Estimation and Hypothesis Testing*, Academic Press, Amsterdam; Boston, Third Edition, 2012.

39. Huber, P. J. and E. M. Ronchetti, *Robust Statistics*, John Wiley & Sons, New Jersey, USA, Second Edition, 2009.

APPENDIX A: APPLICATION

A.1. Base Effective Algorithm

A.1.1. 1. Setup and Video Initialization

```
clc;  clear;

% --- Choose an input clip ------------------------------------------------
[filename1, pathname1] = uigetfile('*.*', 'Select the video');
videoObj  = VideoReader(fullfile(pathname1, filename1));
numFrames = videoObj.NumFrames;

% --- Output container ----------------------------------------------------
outputVideo = VideoWriter('watermarked_video_LL' , 'MPEG-4');
open(outputVideo);
```

A.1.2. 2. Watermark Generation & Global Parameters

```
% 8×8 eye OR checkerboard (choose one)
watermark = eye(8);                    % Diagonal pattern
% watermark = zeros(8);

watermark(1:2:end,1:2:end)=1;

watermark(2:2:end,2:2:end)=1;

WMO      = double(watermark);          % Cast to double
N        = 2;                          % 2-level DWT
L        = 2^N;                        % Sub-sampling factor
wavetype = 'bior6.8';
```

```
% Bi-orthogonal 6.8 (symmetric reconstruction)
K        = 2;                          % Embedding strength

% Pre-allocate result matrices (10×frames for Monte-Carlo)
NCC_i  = zeros(10, numFrames);
PSNR_i = zeros(10, numFrames);

% Per-run buffers
NCC_LL_watermarked = zeros(1, numFrames);
PSNR_values        = zeros(1, numFrames);
MSE_values         = zeros(1, numFrames);
[Mwmo,Nwmo]        = size(watermark);
```

A.1.3. 3. Embedding Loop (10 Monte-Carlo Iterations)

```
for iteration = 1:10                                % Monte-Carlo repeats
    key = 1001; rng(key*iteration);                % Pseudo-random seed
    tic                                             % Time the whole run
    for frameNum = 1:numFrames
        % -- A. Acquire current frame and isolate Cb ---------------------
        frameRGB   = read(videoObj,frameNum);
        frameYCbCr = rgb2ycbcr(frameRGB);
        CbChannel  = frameYCbCr(:,:,2);
        coverimage = double(CbChannel);  [Mc,Nc] = size(coverimage);

        % -- B. Generate LL PN sequence ----------------------------------
        pn_LL = round( 2*(rand(Mc/L, Nc/L)-0.5) );

        % -- C. 2-level DWT ----------------------------------------------
        dwtmode('per');
```

```
[C1,S1] = wavedec2(coverimage,N,wavetype);
cA1     = appcoef2(C1,S1,wavetype,N);
[cH1,cV1,cD1] = detcoef2('all',C1,S1,N);

% -- D. Embed checker / eye pattern into LL ----------------------
wmvector = reshape(WMO,Mwmo*Nwmo,1);
for i = 1:length(wmvector)
    if wmvector(i)==0, cA1 = cA1 + K*pn_LL; end
    pn_LL = round( 2*(rand(Mc/L,Nc/L)-0.5) );   % Refresh PN
end

% -- E. Inverse DWT reconstruction -------------------------------
%    (assemble back cA1,cH1,cV1,cD1 into C1 vector)
x = size(cA1,1); y = size(cA1,2);
C1(1:x*y*4) = [cA1(:).' cH1(:).' cV1(:).' cD1(:).'];
watermarked_Cb = waverec2(C1,S1,wavetype);
watermarked_Cb_uint8 = uint8(watermarked_Cb);

% -- F. Re-insert Cb; convert to RGB -----------------------------
frameYCbCr(:,:,2) = watermarked_Cb_uint8;
watermarkedRGB = ycbcr2rgb(frameYCbCr);

% ==  Fault-injection section  ====================================
%    1) Add Gaussian white noise (variance 1000)
watermarkedRGB = imnoise(watermarkedRGB,'gaussian',0,1000/255^2);

%    2) Every 5th frame blackout
if mod(frameNum,5)==0
    watermarkedRGB(:,:,:) = 0;
end
% ================================================================
```

```
        writeVideo(outputVideo, watermarkedRGB);

        % -- G. PSNR / MSE ------------------------------------------------
        mse = mean( (double(frameRGB(:)) - double(watermarkedRGB(:))).^2 );
        PSNR_values(frameNum) = 10*log10((255^2)/mse);
        MSE_values(frameNum)  = mse;
        PSNR_i(iteration,frameNum)=PSNR_values(frameNum);

        % -- H. Extraction to compute NCC ---------------------------------
        exYCbCr = rgb2ycbcr(watermarkedRGB);
        exCb    = double(exYCbCr(:,:,2));
        [C2,S2] = wavedec2(exCb,N,wavetype);
        cA2     = appcoef2(C2,S2,wavetype,N);

        wm_rec  = ones(1,Mwmo*Nwmo);
        pn_LL   = round( 2*(rand(size(cA2)/L)-0.5) );

        for i = 1:length(wm_rec)
            corrLL(i) = corr2(cA2,pn_LL);
            pn_LL     = round( 2*(rand(size(cA2)/L)-0.5) );
        end
        wm_rec = reshape(wm_rec,Mwmo,Nwmo);
        T_LL   = 2.5*mean(corrLL);

        wm_rec(corrLL > T_LL) = 0;      % Decision rule
        NCC_LL_watermarked(frameNum) = ...
            sum(sum(WMO.*wm_rec)) / (norm(WMO(:))*norm(wm_rec(:)));

        NCC_i(iteration,frameNum) = NCC_LL_watermarked(frameNum);
    end % frame loop
    iterationTime = toc;   % For runtime stats
end % Monte-Carlo loop
```

A.1.4. 4. Finalisation & Visualisation

```
close(outputVideo);

% -------- Summary plots ------------------------------------------------
avg_PSNR = mean(PSNR_values);
PSNR_min = floor(avg_PSNR/5)*5;
PSNR_max = ceil(avg_PSNR/5)*5;

figure; plot(NCC_LL_watermarked,'r','LineWidth',2);
ylim([0 1]); grid on;
title('Frame-wise NCC (LL band)');

figure; plot(PSNR_values,'b','LineWidth',2);
ylim([PSNR_min PSNR_max]); grid on;
title('Frame-wise PSNR');

figure; plot(MSE_values,'g','LineWidth',2);
title('Frame-wise MSE'); grid on;
```

Notes.

- The script now runs *ten Monte-Carlo iterations*; averaging and standard-deviation statistics are stored in `PSNR_i` and `NCC_i`.
- Gaussian noise with variance 1000 and a frame blackout fault (every 5th frame) emulate harsh channel conditions.
- Runtime for each repetition is captured with `tic`/`toc` to evaluate computational cost.

A.2. HMAC Generation

The following MATLAB code shows the generation of the HMAC:

```
% === Example Usage ===
key = 'your_secret_key';
message = 'your_message';
method = 'SHA-256'; % Options: 'SHA-1', 'SHA-256', 'SHA-384', 'SHA-512'

% Compute HMAC hash
hash = HMAC_function(key, message, method);

% Display result
disp(['HMAC Output: ', hash]);


% === HMAC FUNCTION ===
function hash = HMAC_function(key, message, method)
    % Computes the HMAC of a message using the given key and hash method.
    % Inputs:
    %   key    - Secret key as string
    %   message - Message to authenticate
    %   method - Hash method: 'SHA-1', 'SHA-256', 'SHA-384', 'SHA-512'
    % Output:
    %   hash   - HMAC value in hexadecimal string

    % --- Step 1: Input Validation ---
    if nargin < 3
        error('Not enough input arguments. Provide key,...
        ...message, and method.');
```

```
end

% --- Step 2: Determine Block Size Based on Hash Method ---
if any(strcmp(method, {'SHA-1', 'SHA-256'}))
    Blocksize = 64;   % Block size in bytes
elseif any(strcmp(method, {'SHA-384', 'SHA-512'}))
    Blocksize = 128;
else
    error('Invalid method. Use ''SHA-1'', ''SHA-256'',...
    ...''SHA-384'', or ''SHA-512''.');
end

% --- Step 3: Process Key to Fit Block Size ---
if length(key) > Blocksize
    % If key is too long, hash it and pad
    Opt.Method = method; Opt.Format = 'uint8'; Opt.Input = 'bin';
    hashedKey = DataHash(uint8(key), Opt);
    for i = length(hashedKey):Blocksize
        hashedKey(1, i) = 0;
    end
    key_bin = uint82bin8(hashedKey);
elseif isempty(key)
    % Empty key: fill with zero binary cells
    key_bin = repmat({zeros(1,8)}, 1, Blocksize);
else
    % Shorter or exact-size key: convert to binary and pad if needed
    key_bin = str2bin8(key);
    for j = length(key)+1:Blocksize
        key_bin{j} = zeros(1,8);
    end
end
```

```
    % --- Step 4: Define Inner and Outer Padding ---
    i_pad = [0 0 1 1 0 1 1 0]; % 0x36
    o_pad = [0 1 0 1 1 1 0 0]; % 0x5C
    i_pad_bin = repmat({i_pad}, 1, Blocksize);
    o_pad_bin = repmat({o_pad}, 1, Blocksize);

    % --- Step 5: XOR Key with Pads ---
    i_pad_key_bin = bit8xor(key_bin, i_pad_bin);
    o_pad_key_bin = bit8xor(key_bin, o_pad_bin);

    % --- Step 6: Convert to uint8 for Hashing ---
    i_pad_key_uint8 = hex2uint8(bin82hex(i_pad_key_bin));
    o_pad_key_uint8 = hex2uint8(bin82hex(o_pad_key_bin));

    % --- Step 7: Compute Inner Hash ---
    inner_input = [i_pad_key_uint8, uint8(message)];
    Opt.Method = method; Opt.Format = 'uint8'; Opt.Input = 'bin';
    inner_hash = DataHash(inner_input, Opt);

    % --- Step 8: Compute Final HMAC Hash ---
    final_input = [o_pad_key_uint8, inner_hash];
    Opt.Format = 'HEX'; % Final output as HEX string
    hash = DataHash(final_input, Opt);
end


% === Helper Functions ===

% Converts ASCII string to binary cell array of 8-bit vectors
function bin = str2bin8(str)
    bin = cell(1, length(str));
    for i = 1:length(str)
```

```
        b = dec2bin(uint8(str(i)), 8) - '0';
        bin{i} = b;
    end
end

% Converts binary cell array (8-bit) to hexadecimal
function hex = bin82hex(bin)
    hex = cell(1, length(bin));
    for i = 1:length(bin)
        binStr = num2str(bin{i});
        binStr = binStr(~isspace(binStr));
        hex{i} = dec2hex(bin2dec(binStr), 2);
    end
end

% XOR for binary cell arrays
function out = bit8xor(a, b)
    if length(a) ~= length(b)
        error('bit8xor: Input arrays must be of equal length');
    end
    out = cell(1, length(a));
    for i = 1:length(a)
        out{i} = xor(a{i}, b{i});
    end
end

% Converts hex cell array to uint8 array
function out = hex2uint8(hex)
    out = uint8(zeros(1, length(hex)));
    for i = 1:length(hex)
        out(i) = hex2dec(hex{i});
    end
```

```
end

% Converts uint8 vector to binary 8-bit cell array
function bin = uint82bin8(vec)
    bin = cell(1, length(vec));
    for i = 1:length(vec)
        bin{i} = dec2bin(vec(i), 8) - '0';
    end
end
```