

Article

Security First, Safety Next: The Next-Generation Embedded Sensors for Autonomous Vehicles

Luís Cunha ¹, João Sousa ¹, José Azevedo ², Sandro Pinto ¹ and Tiago Gomes ^{1,*}

¹ Centro ALGORITMI/LASI, Escola de Engenharia, Universidade do Minho, 4800-058 Guimarães, Portugal; id11207@alunos.uminho.pt (L.C.); id10564@alunos.uminho.pt (J.S.); sandro.pinto@dei.uminho.pt (S.P.)

² Bosch Car Multimedia, 4705-820 Braga, Portugal; joseazevedo.goncalves@pt.bosch.com

* Correspondence: mr.gomes@dei.uminho.pt

Abstract: The automotive industry is fully shifting towards autonomous connected vehicles. By advancing vehicles' intelligence and connectivity, the industry has enabled innovative functions such as advanced driver assistance systems (ADAS) in the direction of driverless cars. Such functions are often referred to as cyber-physical features, since almost all of them require collecting data from the physical environment to make automotive operation decisions and properly actuate in the physical world. However, increased functionalities result in increased complexity, which causes serious security vulnerabilities that are typically a result of mushrooming functionality and hence complexity. In a world where we keep seeing traditional mechanical systems shifting to x-by-wire solutions, the number of connected sensors, processing systems, and communication buses inside the car exponentially increases, raising several safety and security concerns. Because there is no safety without security, car manufacturers start struggling in making lightweight sensor and processing systems while keeping the security aspects a major priority. This article surveys the current technological challenges in securing autonomous vehicles and contributes a cross-layer analysis bridging hardware security primitives, real-world side-channel threats, and redundancy-based fault tolerance in automotive electronic control units (ECUs). It combines architectural insights with an evaluation of commercial support for TrustZone, trusted platform modules (TPMs), and lockstep platforms, offering both academic and industry audiences a grounded perspective on gaps in current hardware capabilities. Finally, it outlines future directions and presents a forward-looking vision for securing sensors and processing systems in the path toward fully safe and connected autonomous vehicles.

Keywords: autonomous driving; driver-less systems; X-by-wire; security; safety



Academic Editors: Chengjia Han and Zheng Tong

Received: 17 April 2025

Revised: 20 May 2025

Accepted: 22 May 2025

Published: 27 May 2025

Citation: Cunha, L.; Sousa, J.; Azevedo, J.; Pinto, S.; Gomes, T. Security First, Safety Next: The Next-Generation Embedded Sensors for Autonomous Vehicles. *Electronics* **2025**, *14*, 2172. <https://doi.org/10.3390/electronics14112172>

Copyright: © 2025 by the authors. Licensee MDPI, Basel, Switzerland. This article is an open access article distributed under the terms and conditions of the Creative Commons Attribution (CC BY) license (<https://creativecommons.org/licenses/by/4.0/>).

1. Introduction

The automotive industry is currently undergoing a significant transformation as it shifts into the development and deployment of more intelligent connected vehicles [1,2]. This shift aims to equip cars with advanced perception and decision systems, enabling them to effectively perceive and act according to their occupants and surrounding environment. However, to achieve higher levels of automation, modern vehicles require the utilization of robust and complex multi-sensor systems [3,4]. Current vehicles already feature several sensors connected to dedicated Electronic Control Unit (ECU)s, which operate independently of each other and communicate through specialized buses, as depicted in Figure 1. Additionally, although previously limited to functions like motor, lighting, or

door controls, these systems are starting to spread to all domains of the vehicle, including critical operations such as braking and steering control [2,5].

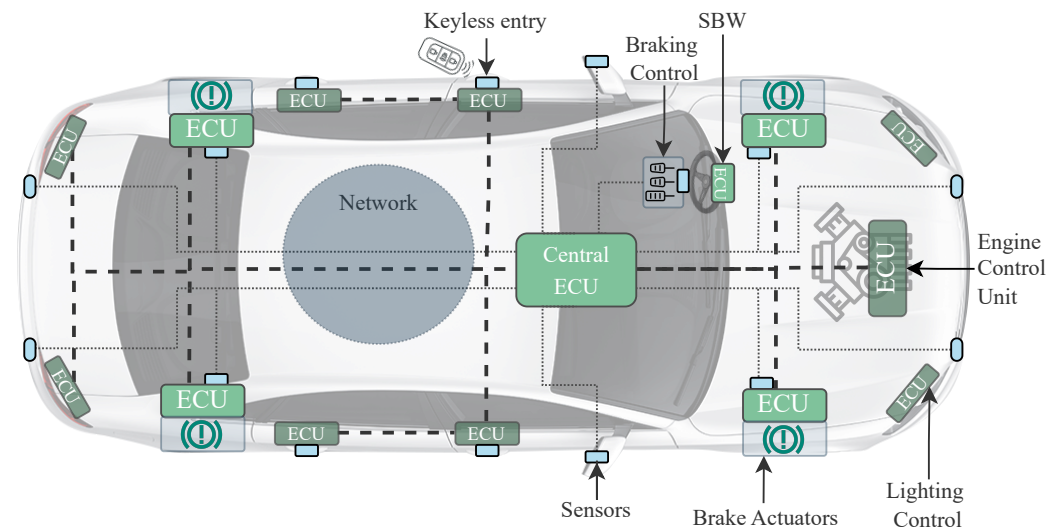


Figure 1. Multiple ECUs across the vehicle.

These systems represent a significant safety risk if they cease to work while in operation, giving rise to concerns regarding the computerization of modern vehicles. To aggravate these concerns, current systems suffer from serious security vulnerabilities, which are typically a result of mushrooming functionality and complexity [6,7]. Two examples of this are the Jeep Cherokee and Tesla Model X hacks, which demonstrate how security vulnerabilities can compromise the safety of automotive systems and their users [8,9]. Given the expected increase in system complexity and integration, safety concerns become meaningless without first ensuring the fundamental pillars of security: confidentiality, integrity, and availability.

To properly address the strict security requirements, modern automotive systems must be devised with a complete chain of trust in place. However, conventional security assurance methods such as encryption tend to be computationally demanding [10]. This poses a significant challenge in the automotive field, as most ECUs are very limited in terms of memory and computational power. Specialized hardware security primitives can mitigate this issue by offloading cryptographic functions, reducing the strain on ECUs' limited resources, and ensuring robust security measures are in place. However, an analysis of current ECU automotive solutions on the market reveals that existing ECU architectures often lack robust hardware security primitives or have none at all.

Additionally, assessing the full scope of possible threats becomes a monumental task, which may leave attack surfaces unintentionally exposed. Stealing secrets by observing specific hardware properties is not a new endeavor [11]; however, it recently gained significant attention because of the side-channel attacks Spectre [12] and Meltdown [13]. Despite the already existent literature about these topics, e.g., cache side-channel attack to infer the location of a vehicle [14], influence automotive rated sensors with out-of-spec stimuli [15], little research about other attack vectors has been made in the automotive field. This suggests that more significant vulnerabilities may remain undiscovered.

Finally, ensuring the availability of an ECU is paramount, both for security and safety. Reliability is one of the most important requirements in the automotive industry due to the associated high safety concerns [16,17]. With inspiration from the avionics industry, reliability is typically achieved by replicating the entire hardware logic, i.e., dual and triple modular redundancy [16]. However, these methods have become prohibitively expensive due to the presence of dozens of ECUs with increasingly complex hardware platforms.

Technologies such as virtualization arise as a cost-effective alternative as they enable the replication of virtual environments by providing underlying primitives for time and space partitioning [18,19].

The increasing need to ensure the confidentiality, integrity, and availability (CIA) triad in automotive systems, together with the ongoing trend towards more complex and connected systems, is creating several security challenges. Consequently, manufacturers are starting to struggle to respect the strict size, weight, power, and cost (SWAP-C) requirements of the industry, while implementing robust security measures. As such, this paper contributes to the literature by: (1) providing a comprehensive analyzes on current automotive ECU architectures; (2) analyzing the use of hardware security primitives in current ECUs, and how they can present a solution for lightweight cryptography; (3) an overview on side channel attacks and how they can be used to target automotive ECUs; (4) a review of current reliability challenges and possible solutions; (5) finally, we look forward, and provide insights on how both research and industry can step towards better solutions.

2. Automotive Sensors: The Big Picture

Current forecasts indicate a significant rise in the sale of autonomous vehicles from 2019 to 2030, with projections suggesting a substantial increase in global sales. By 2030, it is anticipated that approximately 58 million units of vehicles featuring at least Level 3 autonomy will be sold worldwide. Concurrently, the market for automotive sensors is set for notable growth, nearly doubling in size between 2020 and 2025, with estimates reaching around 55 billion U.S. dollars globally [20].

This growth trajectory is in line with the broader expansion anticipated across the automotive electronics industry. This is largely attributed to the substantial shifts occurring in vehicle architectures, primarily driven by vehicle electrification, growing emphasis on advanced driver assistance systems (ADAS), and autonomous features, as well as vehicles' increasing connectivity. Of particular note is the automotive semiconductor market, which, as Figure 2 demonstrates, is expected to experience significant growth from 53 billion U.S. dollars in 2021 to approximately 103 billion U.S. dollars by 2029 on a global scale. This growth emphasizes the role played by semiconductor technology in advancing the capabilities and functionalities of modern vehicles.

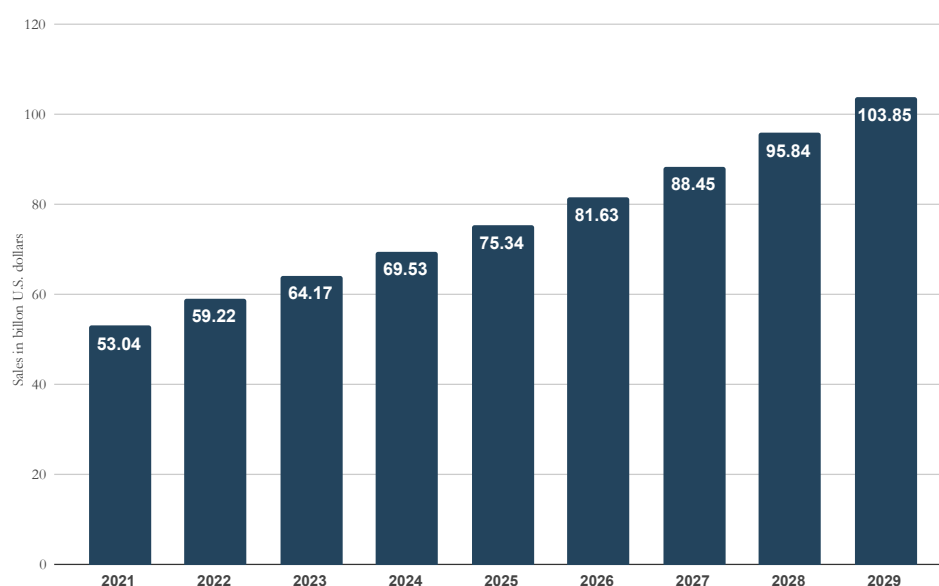


Figure 2. Size of the global automotive semiconductor market in 2021, with a forecast until 2029.

2.1. Automotive Architecture

Modern automotive architectures often comprise numerous sensors linked to dedicated ECUs. These ECUs are dispersed throughout the vehicle, communicating via specialized communication buses, and manage a range of tasks from engine control to navigation. They are typically categorized into domains such as powertrain, chassis, body, infotainment, and telematics, based on their functions [21]. Moreover, they can be further subdivided into subsystems responsible for specific vehicle features. In the past, these subsystems were simple, consisting of only a few basic components. However, the industry's current trend favors high system integration and functionality through complex multi-sensor and multi-ECU setups. For instance, Figure 3 illustrates the evolution of steering system architecture over the years. Figure 3a presents a conventional power-assisted steering system, while Figure 3b depicts a potential steer-by-wire (SBW) configuration [22]. This growth in complexity inevitably introduces new security and safety challenges that need to be addressed.

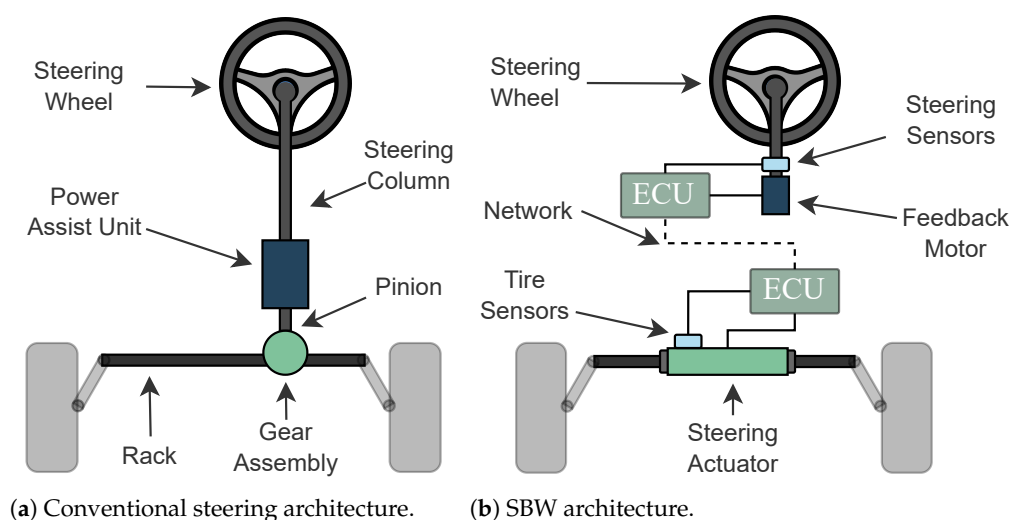


Figure 3. Steering architectures.

Understanding the criticality associated with each domain is important to assure the deployed solutions are not only robust and safe but also adhere to strict SWAP-C requirements. Central to this understanding is the concept of automotive safety integrity level (ASIL), which serves as a risk classification system outlined in the ISO 26262 standard [23] for ensuring the functional safety of road vehicles. The ASIL classification system categorizes risks into four groups: ASIL A, B, C, and D. ASIL A denotes the lowest level of automotive hazard, while ASIL D indicates the highest. The following list highlights the major tasks and possible ASIL categories of the previously identified domains:

- **Powertrain:** ECUs in this domain are responsible for controlling components that generate the power necessary for the car to move. They usually handle tasks regarding to engine control, e.g, throttle-by-wire, transmission control, battery management. A failure in one of these subsystems can represent a severe safety risk such as unintentional acceleration in the car. As such, systems associated with this domain typically have ASIL-D certification [24,25].
- **Chassis:** ECUs in this domain oversee the control of several chassis components, including wheels, brakes, and suspension. Tasks within this domain encompass critical functionalities such as anti-lock braking system (ABS), brake-by-wire (BBW), and SBW. A malfunction in any of these subsystems can pose a significant safety risk, such as the vehicle failing to respond to steering commands. Consequently, systems

within this domain typically boast ASIL-D certification to ensure their reliability and safety [26,27].

- **Body:** This subsystem is notably diverse, encompassing a wide range of functionalities. It includes passive safety tasks like airbag deployment and seat-belt monitoring, as well as comfort and convenience features such as central locking systems, heating, and parking assistance. The ASIL classifications within this domain vary significantly depending on the specific subsystem. For instance, airbag systems typically boast ASIL-D certification [28], whereas heating systems may only necessitate ASIL-B certification.
- **Infotainment and telematics:** Subsystems within this domain manage a variety of features, including audio control, video, smart display, Global Positioning System (GPS) navigation, and wireless communication. Typically, these subsystems only require ASIL certifications up to level B [29]. Precise vehicle positioning and environmental perception are critical system requirements for autonomous driving. Recent advancements in ADAS have increased the need for high-accuracy absolute positioning [30], typically achieved through global navigation satellite systems (GNSS), as well as relative positioning between vehicles using vision or LiDAR sensors [31]. Given their role in real-time decision making, these systems may require certification up to ASIL-D [32]. Moreover, ensuring the integrity and authenticity of spatiotemporal reference data is increasingly important to protect against spoofing, jamming, or malicious manipulation of localization inputs.

A growing trend in the automotive industry is the deployment of mixed-criticality systems (MCS). These are embedded or real-time systems that execute workloads with different levels of criticality, such as safety-critical and non-safety-critical tasks, on a shared hardware platform. For instance, modern vehicles often combine network-connected infotainment systems with safety-critical control systems like anti-lock braking. This integration brings major design challenges, particularly in ensuring system safety, real-time responsiveness, timing predictability, and freedom from interference (FFI) [33]. The ISO 26262 functional safety standard reinforces this concern by requiring that software components with different ASIL ratings operate without interfering with each other.

Interference between ASIL components constitutes a violation of safety requirements. This can happen when a component affects another of higher, equal, or even lower ASIL. To prevent such violations and ensure FFI, the system must provide strong spatial and temporal isolation [34]. Spatial isolation ensures that physical resources, such as CPUs and memory, allocated to one component remain inaccessible to others. Temporal isolation guarantees that the timing behavior of one component's execution does not disrupt another's. However, achieving both forms of isolation remains challenging due to contention on shared microarchitectural resources. Components often compete for access to elements like the last-level cache, main memory, and the system bus. This contention increases execution times and reduces system determinism, making it particularly difficult to guarantee timing predictability for hard real-time systems.

On top of the safety standards, adhering to conventional security requirements remains imperative in the automotive domain. These requirements typically include: (1) restricting access to security-critical data to authorized entities only; (2) ensuring that security-critical data remains unaltered by unauthorized entities; (3) ensuring critical services operate within specified time constraints; (4) enabling ECUs to verify each other's identities and restrict resource access accordingly. Assuring these requirements is very dependent on the utilized ECUs' and vehicle electronics' architectures.

2.2. ECU Architecture

The typical design of contemporary ECUs comprises several key components [35]. As Figure 4 demonstrates, these components include a sensor module, a computation and control module, a communication interface, and sometimes, an actuator component. Within this setup, each ECU serves the dual function of both supplying and receiving information within the vehicle's network while also responding to external data inputs. The acquisition module is primarily responsible for interfacing with sensors, retrieving sensor data, and potentially performing preprocessing tasks. Meanwhile, the communication interface is responsible for interfacing with the vehicle's network, ensuring seamless communication between ECUs and other vehicle components. The computation unit, typically equipped with a microcontroller unit (MCU), plays a central role in processing data collected from connected sensors and controlling relevant actuators. Given the critical role of the MCU, its characteristics are of paramount importance, typically including extensive peripheral support and specifications tailored for automotive applications, which ensure reliability in demanding automotive environments [35].

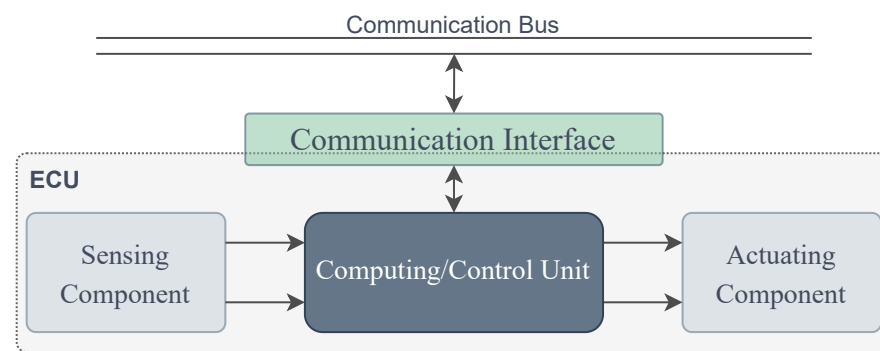


Figure 4. ECU architecture diagram.

The architecture of a specific ECU is largely proprietary, determined by the original equipment manufacturer (OEM), and not publicly disclosed. However, some assumptions can be made to better understand its components. The first assumption is that the MCU can be considered the primary component of the ECU, as many of its requirements and constraints are directly linked to it. Nonetheless, detailed information about the most commonly used types of MCUs is often not readily available. However, the second assumption is that, by studying the market dynamics within this field the types of MCUs families utilized by the automotive industry can be deduced. Various manufacturers offer automotive-rated MCU solutions, each with different performance ratings, power consumption levels, and security features. Figure 5 provides an overview of the automotive semiconductor industry in 2021, focusing on the market shares of the top 5 manufacturers in this sector, including their MCU manufacturing market share [36]. The leading vendors in this industry, ranked by market share from highest to lowest, include Infineon Technologies, NXP Semiconductors, Renesas Electronics, Texas Instruments Inc., and STMicroelectronics. While the presented market shares encompass all semiconductor products and not just MCUs, cross-analyzing this data with the MCU manufacturing market offers valuable insights. The top 5 semiconductor providers in the automotive field collectively hold more than 70% of the market share in the MCU segment for the year 2021. Therefore, the final assumption is that the potential MCU architectures employed in current ECUs are based on the product portfolios offered by these manufacturers. Table 1 highlights some key findings from this analysis.

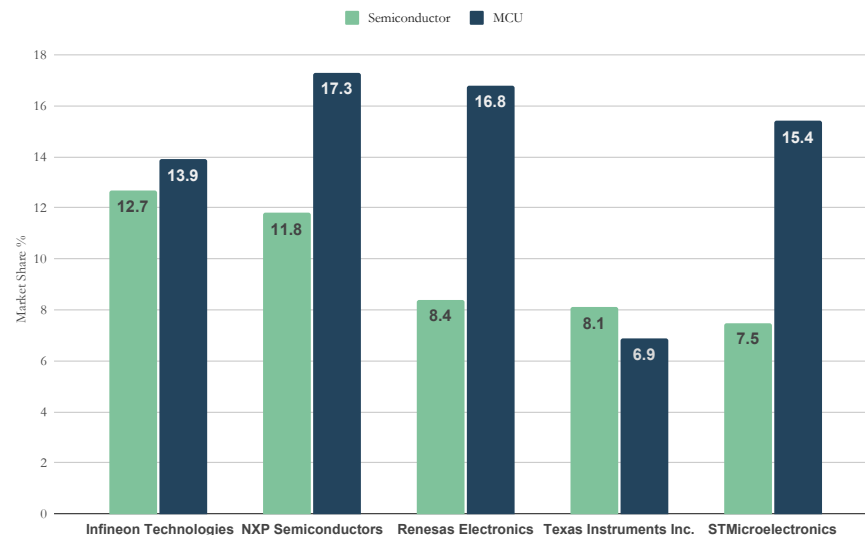


Figure 5. Automotive semiconductor vendor 2021 market shares.

The top vendor by market share, Infineon Technologies, offers multiple solutions targeting the various aspects of the vehicle. Currently, most Infineon Technologies' products, suitable for automotive sensor/ECU applications, use the Armv6-M and Armv7-M architectures. Additionally, NXP Semiconductors offers ten line-ups of automotive products that use multiple Arm architectures and five line-ups using various other architectures, e.g., Power Architecture. Almost all of Renesas Electronics' SoC solutions for automotive use Armv7-m architectures, with some based on their V850 architecture. Regarding Texas Instruments Inc., half of its automotive repository is based on the Armv7-R architecture and the other half on the C2000 architecture. Finally, STMicroelectronics has a flagship family of MCUs for automotives featuring the Armv8-R architecture, and lower entry MCUs featuring PowerArchitecture and their own ST architecture. Although it is not possible to claim the exact percentages associated with the architectures used in current automotive ECUs, it can be inferred that the most used architecture is Arm (more specifically Armv7-M) by a relatively large margin, considering the top vendors' portfolios.

Table 1. Automotive semiconductor vendor's MCU architectures. ✓—Architecture supported.

MCU Families	ARM	PowerPC	V850
Infineon Technologies	✓		
NXP Semiconductors	✓	✓	
Renesas Electronics	✓		✓
Texas Instruments Inc.	✓		
STMicroelectronics	✓	✓	

3. Securing Automotive Systems: The Role of Hardware Security Primitives

As the automotive industry shifts towards a more computerized and interconnected environment, prioritizing security becomes increasingly crucial. However, traditional security measures such as encryption and authentication often come with significant computational overhead [10]. Moreover, as demonstrated by the most used MCU architectures, ECUs typically exhibit limitations in terms of memory and computational power. To address these challenges, specialized hardware security primitives emerge as a potential

solution, as they can be utilized to offload heavy security algorithms to dedicated hardware components. These primitives, encompassing a range of specialized hardware components and technologies, offer robust solutions for tackling the mentioned requirements. One way to enforce the requirements 1, 2, and 3 is by making it impossible for unauthorized parties to access specific memory regions or critical hardware components. Memory isolation technologies, such as Arm memory protection unit (MPU) and RISC-V physical memory protection (PMP), serve as hardware security primitives designed to safeguard specific memory regions. These mechanisms allow for the definition of access permissions for individual memory regions, enabling the implementation of strict access control policies. Through these measures, unauthorized entities are effectively prevented from accessing critical data or tampering with sensitive code segments.

Likewise, system-wide isolation technologies, e.g., Arm TrustZone, SiFive WorldGuard (WG), can also be used. As demonstrated in Figure 6, these technologies operate by creating distinct hardware domains, effectively partitioning the computing resources of a device into multiple execution environments. Each execution environment operates independently from each other, preventing unauthorized entities from accessing sensitive data or interfering with critical system functions. However, despite the strong security benefits offered by system-wide isolation technologies, their adoption in the lower-end MCU market has been relatively limited until recently. The introduction of the Armv8-M architecture marks a significant milestone in this regard, as it introduces a variant of TrustZone technology to the Cortex-M family of microcontrollers [37]. A similar strategy involves utilizing isolated secure coprocessors, which establish a physically secure subsystem (completely independent) typically resistant to tampering and equipped with cryptographic functionalities. In addition to the previous requirements, these modules can also be used to satisfy requirement 4 by enabling ECUs to use more robust cryptographic functions for device authentication. For example, a secure element is a chip which provides a secure and protected environment for both applications and data, effectively safeguarding them from unauthorized access and side-channel attacks. Similarly, a trusted platform module (TPM) functions as a secure crypto-processor tasked with generating and safeguarding cryptographic keys and security critical data.

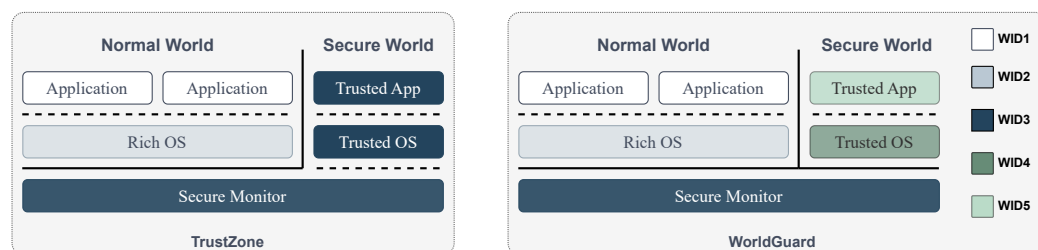


Figure 6. Representation of TrustZone and WorldGuard functioning.

Nonetheless, no requirement can be met without a proper root of trust (RoT). A hardware RoT is the foundation on which all secure operations of a computing system depend. It usually contains the keys used for cryptographic functions and enables a secure boot process. As it must be secure by design, a RoT is typically implemented in hardware, making it immune from malware attacks [38]. There are several technologies that allow the creation of an RoT such as one-time programmable (OTP) memories, read-only memories (ROM), physical unclonable function (PUF), and more. OTP memories and ROM are types of memory that can only be written once, making them ideal for secure storage of security-critical data. On the other hand, PUFs are microchips capable of generating a unique “digital fingerprint” through their inherently random microstructure. This distinctiveness arises from uncontrollable and unpredictable physical variables introduced

during the manufacturing process, making them impossible to replicate [39]. Another component widely used as a RoT is a hardware security module (HSM). A HSM is a dedicated and tamper-resistant hardware device designed to provide secure key management and cryptographic operations. A project worth mentioning in this area is the EVITA project, where a specification for an automotive HSM was created and implemented [40]. Table 2 summarizes some security-oriented technologies and how they can be leveraged for enhanced security.

Table 2. Security-oriented techniques and hardware security primitives. ✓—Primitive supported.

Primitive/ Techniques	Encryption	Memory Isolation	Hardware RoT	Secure Coproprocessors	Cryptographic Accelerators	TEE	Secure Boot
MPU/PMP		✓					
TrustZone/WG		✓				✓	✓
Secure Elements	✓	✓	✓	✓	✓		✓
TPM	✓	✓	✓	✓	✓		✓
HSM	✓		✓		✓		✓
RNG	✓				✓		
OTP			✓				✓
ROM			✓				✓
PUF			✓				✓

3.1. Use Case Example

As highlighted, contemporary automotive architectures integrate numerous ECUs interconnected via specialized communication buses. The communication among ECUs presents a prime use case scenario for leveraging security hardware primitives. Figure 7 illustrates a potential secure system stack, showcasing two separate realms: one secure and one non-secure, facilitating the execution of various security-critical services.

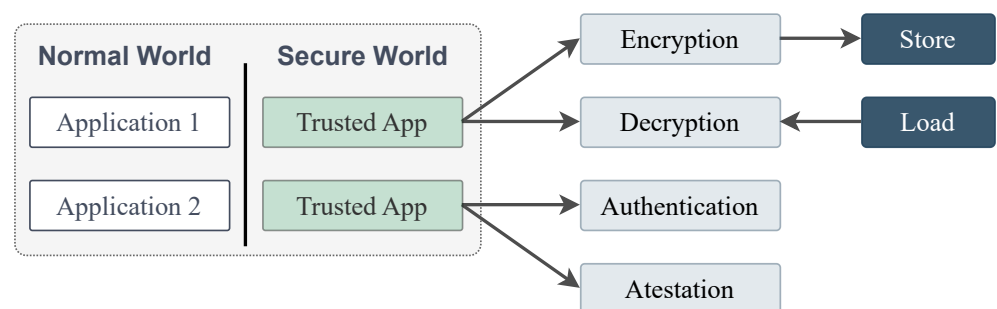


Figure 7. Example of some security-critical services.

The non-secure world manages all regular ECU operations, while the secure world is tasked with handling security-critical communication services such as attestation, authentication, encryption, and decryption of data. Encryption and decryption serve as an essential means to safeguard sensitive information from unauthorized access, whether the data is at rest or in transit. In the presented use case, encryption and decryption can be employed to: (1) securely store data received from other ECUs (data at rest); (2) decrypt data received from other ECUs (data in transit); (3) encrypt critical data before transmitting it over the communication bus (data in transit). Various options exist for implementing cryptographic services: software-based solutions, preferably within a trusted execution environment (TEE), or utilizing cryptographic accelerators. Considering the limited resources of typical ECUs, dedicating processing power to complex cryptographic algorithms

can incur significant computational costs. Therefore, software-based solutions are discouraged in low-power automotive solutions. As such, cryptographic accelerators often emerge as the solution for both on-the-fly decryption and encryption, ensuring minimal performance overhead and latency. As demonstrated in Table 2, some primitives like the HSM, secure element, and TPM can be used to provide dedicated hardware support for cryptographic operations.

Another set of necessary security-critical services includes authentication and attestation, which align with stipulated requirement 4, emphasizing the necessity for ECUs to verify each other's identities. Authentication services play a crucial role in the secure exchange of data by enabling an ECU to sign security-critical information before transmission via the communication bus. This process ensures the integrity and authenticity of the data, safeguarding against unauthorized alterations or malicious tampering. Similarly, attestation serves as a crucial mechanism for verifying the integrity of ECUs within the automotive network. By performing attestation, an ECU can validate that the sender of the data has not been compromised or subjected to unauthorized modifications. This verification process provides assurance regarding the trustworthiness of the originating ECU, generating confidence in the received data's integrity and authenticity. Utilizing a hardware RoT is essential for enabling such services, as it ensures the secure storage and management of cryptographic keys. As demonstrated by Table 2, several mechanisms, such as secure elements, TPM, HSM, OTP, ROM, and PUF, can be used to implement a RoT.

Nonetheless, unless TEE technology alongside a system-wide isolation technique is used, every security-critical component that runs in the software layer, e.g., security drivers, data management, might be exposed to various kinds of side-channel attacks or hijacking [41,42]. TEEs are typically considered more secure than modern operating systems because they rely on hardware-enforced separation and have a significantly smaller trusted computing base (TCB). Consequently, TEEs have gained widespread adoption for protecting mobile devices against malware. More specifically, Arm TrustZone has become the primary hardware technology to implement TEEs in mobile environments [43]. While this section merely presents a use case example illustrating the utilization of hardware security primitives for securing inter-ECU communication, it underscores the critical importance of integrating at least some of the mentioned components. Failing to do so renders meeting the highlighted security requirements considerably challenging, ultimately compromising the capacity to maintain secure, robust, and low-latency security services.

3.2. Currently Used Hardware Primitives

Current portfolios of leading manufacturers already provide automotive products that encompass some of the mentioned hardware primitives. However, some solutions do not include any hardware security features or are often limited in the ones they support. As illustrated in Table 3, an exploration of the top five manufacturers' portfolios for emerging platforms like Armv8-M with Trustzone-M support reveals that the transition to next-generation architectures is still in the early stages. While three of the top five automotive semiconductor vendors already provide solutions integrating MCUs with Armv8-M cores, only one offers an automotive solution leveraging this architecture. This highlights the ongoing adoption of advanced security-focused architectures in the automotive semiconductor industry. Although these primitives provide robust isolation and cryptographic support, their adoption must consider implementation trade-offs in terms of power, latency, and silicon complexity. Secure elements and TPMs offer strong physical security and dedicated key management but often require external components and additional integration effort. TrustZone, while integrated into some Arm-based MCUs, introduces minor context-switching overhead but is limited to higher-end devices. Conversely, MPU-

based TEEs are lightweight and widely supported across automotive-grade MCUs but offer weaker isolation guarantees. Hardware security modules (HSMs) deliver excellent performance for cryptographic operations but tend to increase die area and are generally integrated only in mid- or high-end domain controllers. These trade-offs are critical in low-resource environments where area, power, and real-time performance constraints are tightly coupled.

Table 3. Top 5 automotive semiconductor vendors’ Armv8-M support. ✓—Architecture supported.

Vendor	Armv8-M Support	Armv8-M Automotive Line
Infineon Technologies		
NXP Semiconductors	✓	✓
Renesas Electronics	✓	
Texas Instruments Inc.		
STMicroelectronics	✓	

As a result, to the best of our knowledge, no commercial offering built on top of these platforms supports the incorporation of these new hardware security-oriented technologies [43]. For instance, Vector’s security solution capitalizes on the inherent security features of MICROSAR, encompassing support for established cryptographic protocols and ensuring data security throughout processing, transit, and storage. Nevertheless, MICROSAR’s isolation mechanisms still rely on conventional hardware primitives such as the MPU. While promising solutions have been proposed by academia in [44,45], the former may not be feasible for implementation in resource-constrained ECUs, and the latter was not specifically designed to address the unique requirements of automotive sensors.

Assessment of commercial platforms. An analysis of the top five vendors’ portfolios reveals that current architectures generally fall short of meeting the requirements outlined in our use case. Among them, NXP is the only vendor offering automotive products based on Armv8-M, making the S32Z2 family the sole lineup capable of fully addressing the specified needs. Existing architectures can still leverage solutions like MCU-based TEEs to achieve comparable levels of software compartmentalization, though without full system-wide isolation [46]. For instance, Hexfive’s MultiZone enforces separation between zones by relying on basic hardware primitives, such as the MPU and privilege levels [47]. This approach enables platforms equipped with an MPU and a hardware cryptographic accelerator to provide similar isolation guarantees at the software level. Examples of such product families include Infineon’s TRAVEO, NXP’s S32K3, Renesas’ RH850, and STMicroelectronics’ Stellar, all of which integrate an MPU and secure elements. Nonetheless, the use of MCU-based TEEs introduces undesirable performance overhead and cannot provide the system-wide isolation guarantees that a hardware-assisted TEE technology like TrustZone offers. Such limitations become critical when integrating multiple sensors with differing ASIL requirements and real-time constraints into a single ECU. The lack of robust isolation mechanisms complicates mixed-criticality execution, particularly in safety-critical domains like sensor fusion and perception pipelines. As a result, existing architectures cannot reliably guarantee freedom from interference or meet timing predictability for high-ASIL applications.

3.3. Side-Channel Attacks

Side-channel attacks are a type of attack that allows a bad actor to derive secret values or influence a system using unintentional leaks of data or the system’s behavior [48].

However, microarchitectural side-channels are just the tip of the iceberg, and in the context of automotive systems, there is very little research on the topic. As modern ECUs are usually complex cyber-physical systems, understanding all the possible attack vectors is essential to address all security requirements. Moreover, protecting the actual sensors is mandatory as most applications rely on the accuracy and trustworthiness of their measurements, making it a common practice to inherently trust sensor output [15]. However, the wide range of sensor types and measurement techniques makes it challenging to systematically analyze and defend against sensor-based side-channel attack [49].

One of the most used attack vectors in the literature is electromagnetic radiation. It is widely used to attack sensors because either they are already sensitive to this type of physical phenomena, or because the electronic components inside the target device are susceptible to several forms of interference [49]. This attack vector was already been succeeded in automotive grade sensors such as LiDAR, cameras, and radar. For example, Petit et al. [50] and Shin et al. [51] induced fake dots in an automotive LiDAR by relaying and replaying intercepted LiDAR pulses with a delay. Similarly, Yan et al. [52] proposed the injection of radio waves into a radar sensor, allowing them to successfully jam a radar sensor present in a Tesla vehicle. Likewise, electromagnetic radiation can be used to attack computational systems. However, its main use is to extract information from leaking electromagnetic radiation [53,54]. This side-channel can be exploited when the amount of emitted electromagnetic radiation has a correlation with the operation of the system. For example, Gandolfi et al. [55] managed to extract keys from three distinct MCUs and Camurati et al. [53] demonstrated the issues associated with the use of mixed-signal chips. In these devices, electromagnetic radiation from digital circuits propagate to nearby radio components and are broadcasted unintentionally.

Another very relevant attack vector are acoustic waves. Despite not being used for attacking computing systems, this attack vector is of high relevance when discussing sensory systems. Microelectromechanical systems (MEMS), gyroscopes, and accelerometers measure acceleration and rotation by using a small mass that moves in response to acceleration forces and, because of that, are very sensitive to acoustic waves [49]. By using acoustic waves at very specific frequencies, Trippel et al. [15] were able to perform denial of service (DoS) and spoofing attacks on multiple automotive grade sensors. Additionally, this attack vector has been used to influence ultrasonic sensors in automotive contexts. In the works presented by Yan et al. [52] and Xu et al. [56], multiple automotive grade ultrasonic sensors were both jammed and spoofed. They injected artificially generated ultrasonic signals, making the tested vehicle falsely assume there were obstacles present.

A summary of the most relevant side-channel attack vectors in automotive contexts, their affected components, and applicable countermeasures is presented in Table 4. However, every part of an ECU is potentially vulnerable to side-channel attacks. For instance, the time it takes to perform an operation on a computer can vary based on several factors, including the type of operation, the input data, the hardware architecture and implementation, and the characteristics of the operating environment [57]. Attackers can measure these time differences to understand the internal state of the system and possibly access sensitive data. The most famous timing attacks are the Meltdown [13] and Spectre [12], which exploited the speculative and out-of-order execution of modern computer processors. Although it was thought that these type of attacks were limited to high-end devices, Bulck et al. [58] demonstrated a side-channel attack affecting a wide range of systems, from embedded devices to high-end computers, which can infer execution parameters by measuring the latency of a precisely timed interrupt. Recently, attacks of this nature targeting automotive architectures started to emerge. Luo et al. [14] leveraged time-base cache side-channel attacks to infer an autonomous vehicle's location and trajectory, com-

promising the privacy of its occupants. These attacks exploit the relationship between a vehicle’s physical state and the cache access patterns of its control software. By analyzing these patterns, attackers can deduce the vehicle’s route or location. This vulnerability arises from the adaptive algorithms employed in autonomous vehicles, which dynamically adjust computational tasks based on environmental changes or uncertainties.

Table 4. Side-channel attack vectors and corresponding countermeasures in automotive systems.

Attack Vector	Target Component	Countermeasures
Electromagnetic injection	LiDAR, radar, MCUs	Shielding, signal filtering, protocol-level redundancy
Acoustic injection	MEMS gyroscopes and accelerometers	Mechanical damping, frequency filters, casing design
Timing attacks	MCUs, control software	Constant-time algorithms, TEE isolation, jitter insertion
Cache side-channels	Software stack, control units	Cache partitioning, access pattern obfuscation, TEE support
Bus arbitration leakage	MCU interconnects	Hardware-level redesign, bus access scheduling randomization
Radio interference	Ultrasonic, RF sensors	Spread-spectrum techniques, authentication protocols

Additionally, a novel non-conventional side-channel attack, which targets resource constrained devices, was discovered. It leverages the MCU bus interconnect arbitration logic to bypass hardware memory isolation primitives and leak information from protected and trusted applications. Rodrigues et al. [59] provided evidence about the existence of this side-channel in 11 MCUs from five different ISAs (including the complete Arm Cortex-M family) and five different silicon manufacturers, including NXP and STM. They successfully compromised a smart lock application from Vulcan [60], a vehicular authentication system that adheres to a reference implementation of a trusted keypad provided by Texas Instruments. Although there are well-established countermeasures for most of the sensor attacks discussed [49], this section highlights that cybersecurity in automotive systems must extend beyond simple attack vectors and encompass all system components. Therefore, it is essential to include the implementation of more secure architectures that systematically address each layer of the system. Nonetheless, the full impact of side-channel attacks on ECUs has yet to be determined.

4. Automotive Reliability: Redundancy as the Key to Availability

Reliability problems associated with electrical systems have existed since the first circuit, mainly due to components’ characteristics (e.g., vacuum tubes and relays). However, nowadays these issues predominantly arise from the significantly increased complexity of both hardware and software [61]. With the growing market traction for smaller transistors, higher clock frequencies, and lower core operating voltages, alongside other advancements that increase system complexity, systems have become more susceptible to reliability and security challenges [62,63].

Dependability is the capability of a system to deliver its intended level of service to its users, which involves preventing faults from lasting longer than acceptable or evolving into errors or failures. This property is essential in automotive systems as a failure can seriously compromise the safety of the users. For instance, a real-world example of those threats can

be reflected in a hydraulic braking system. When a hole is in the brake fluid transportation tubes (an instance of a fault), and a user applies pressure to the pedals, hydraulic leakage occurs (an example of error). Subsequently, with insufficient oil remaining in the braking system, the car fails to stop when required, potentially leading to an accident (an instance of failure). To mitigate these threats and ensure system availability, especially under real-time constraints, the system must achieve high levels of dependability. This can be accomplished through fault mitigation strategies such as fault prevention and fault-tolerance techniques [64]. By employing these techniques, systems can detect, identify, and recover from faults or errors, maintaining their functionality and reliability.

Fault prevention. Fault prevention is applied mainly during the development phase, aiming to prevent faults from occurring during the system's runtime execution. This approach focuses on identifying potential sources of faults and implementing techniques to eliminate or reduce the likelihood of these faults happening. Techniques might include robust design, thorough testing, code reviews, quality control, and adherence to standards.

Fault tolerance. Fault tolerance occurs during the system's lifetime. It refers to mechanisms to deal with faults once they manifest, involving strategies that combine error detection and recovery techniques [64]. Error detection is the process of identifying the presence of errors within a system, which can be concurrent [65], where the system performs parallel verification during operation, or periodic [66,67], where the error detection mechanism suspends the system's operation to verify a faulty presence. Complementarily, error recovery mechanisms are strategies and processes used to restore a system or service to a safe and correct operational state after a fault or error occurs. These recovery mechanisms may involve roll-back or roll-forward methods, along with checkpointing techniques that save the system states at specific times [68,69]. Figure 8 illustrates both methods on a timeline. In *Replica 1*, the roll-back method reverts the system state to a prior safe state (time 1). In contrast, in *Replica 2*, the roll-forward method transfers the system state to the current safe state of another healthy replica (time 3).

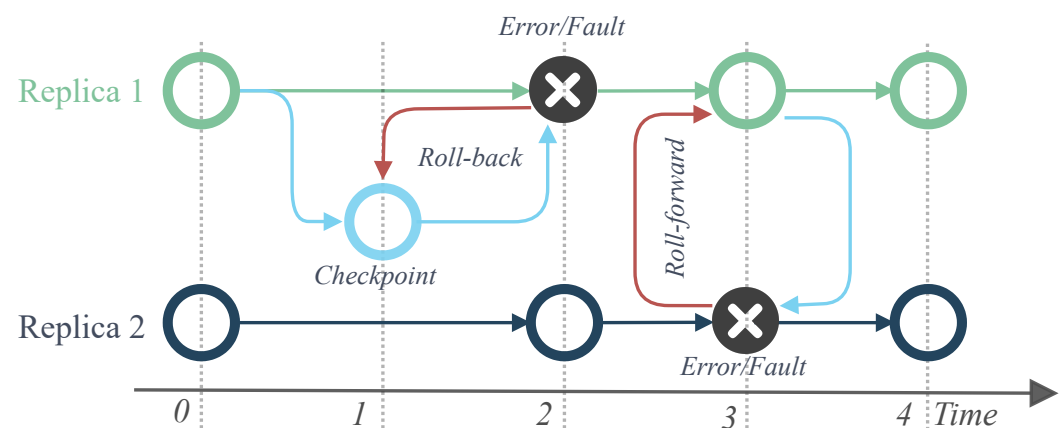


Figure 8. Roll-back and roll-forward system with checkpointing mechanism.

Currently, there are numerous fault-tolerance implementation methods, redundancy techniques being the most used [70]. These consist of the replication of critical software or hardware components within a device or circuit. Redundancy ensures that if one component fails, the redundant one can seamlessly take over, thereby preventing system failure and maintaining the safe state operation. Overall redundancy can be implemented using software-based [71], hardware-based [72,73], or hybrid [69,74] techniques.

Software redundancy. Regarding software-based techniques, various architectural approaches can be employed to implement redundancy. These methods include duplicating

software components into multiple instances of the same code, often leveraging multi-threading techniques or aspect-oriented programming [75]. Two common strategies for implementing software-based redundancy are N-Version Programming [71] and Time-based redundancy [76,77]. The former involves creating multiple equivalent software versions of functionalities, providing higher fault coverage levels. The latter involves executing and comparing critical operations multiple times across different execution times, allowing the identification and comparison of errors. Moreover, unlike these redundancy methods that replicate physical components or entire systems, another form of redundancy for error detection involves duplicated information resources. Redundancy through information often involves using additional bits to ensure data integrity during, for example, the transmission or storage of data. Some of this redundancy through information are parity checks [76] and cyclic code checksums [78].

Hardware redundancy. Regarding hardware redundancy, these methods involve creating several instances of critical hardware components [79]. These redundant components are typically set up in parallel and can be implemented using passive [80], active [81], or hybrid methods [82]. Passive hardware methods tolerate faults by ensuring the system continues operating without depending on corrupted or faulty data, which usually involves a voting system to compare results from all replicas. For example, Figure 9a illustrates the triple modular redundancy (TMR), a passive method that connects three replicas to a voter unit, where the input data is the same across the identical systems and the output is chosen according to the majority. As a result of this voting process, the system can ignore the error occurrence in one of the replicas and continue to operate in a non-faulty state. On the other side, active methods actively detect the presence of a fault in a system.

A popular active redundancy technique is duplication with comparison (DWC), as illustrated in Figure 9b. In this method, the system incorporates dedicated hardware (Comparator) to compare the input data of two replicas and generate an Error Signal when differences are detected [82]. In case of errors, since this signal does not identify the faulty replica, all replicas are typically recovered to a last non-faulty state. While active redundancy enhances fault detection, it lacks the selective recovery capabilities of passive methods. To address this limitation, hybrid hardware redundancy methods integrate elements of both approaches. These techniques are widely used in safety-critical applications, leveraging the real-time fault-masking capabilities of passive methods while incorporating the fault recovery mechanisms of active redundancy. Typically, hybrid approaches identify and restore faulty replicas to a safe state based on correctly functioning replicas, often using roll-forward recovery methods to ensure system integrity.

Hybrid redundancy. Lastly, recent trends are moving towards hybrid redundancy solutions, which combine the benefits of both hardware- and software-based redundancy techniques. Some examples of hybrid solutions are lockstep [83,84] and virtualization-based redundancy methods [74]. Lockstep is a widely used fault-tolerance solution that leverages different cores to execute the same application simultaneously while comparing their outputs and taking further actions in case both behave differently. Two examples of this redundancy technique are the dual-core lockstep (DCLS) and the triple-core lockstep (TCLS). Contrarily to DCLS, the outputs from TCLS are majority-voted [84]. Despite being primarily classified as a hardware solution, recent literature proposes combining additional software-based redundancy strategies to address the high susceptibility to common-mode failures (CMF). A CMF is a type of failure in which multiple systems or components fail similarly due to a shared cause or a single point of failure [85]. For instance, in a vehicle with redundant braking controllers using the same software vulnerable to a latent error, an incorrect braking response will simultaneously cause the entire braking system to fail. Since

all redundant controllers exhibit the same faulty behavior, the system lacks an independent recovery mechanism to mitigate the failure, potentially leading to a loss of braking control and severe safety risks. To solve this, recent literature proposes the use of time diversity [86], microarchitecture diversity [87], ISA diversity [88], and multi-threading [83].

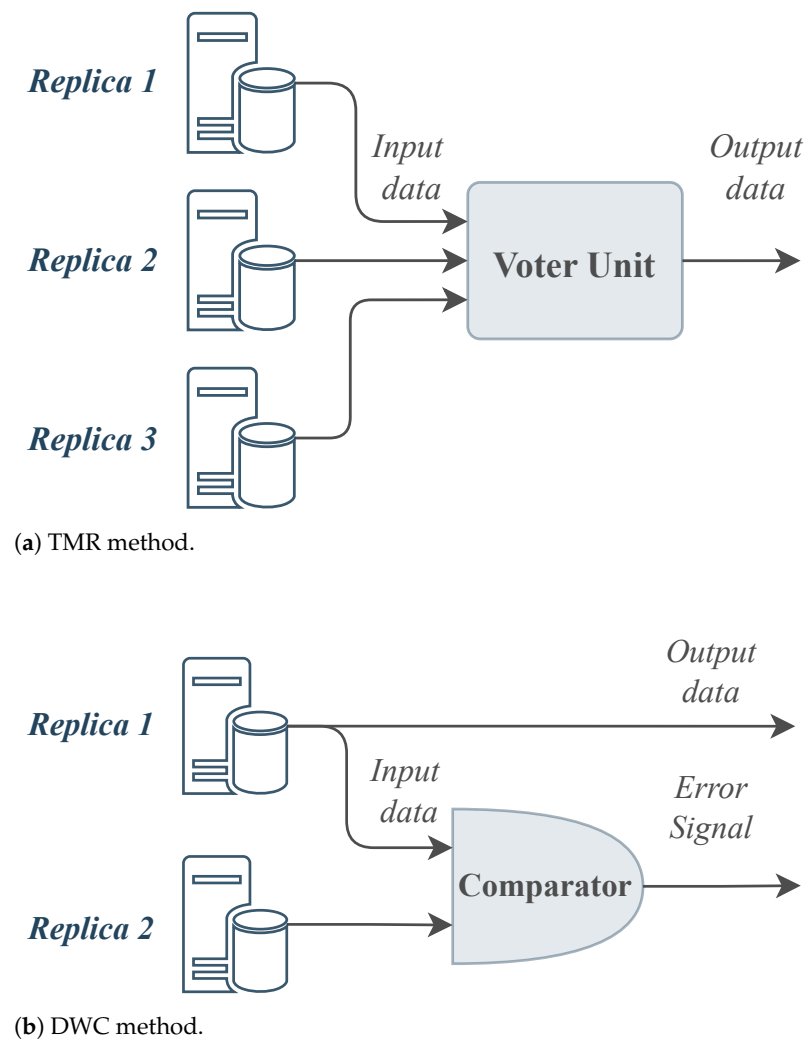


Figure 9. Active and passive hardware fault-tolerant methods.

When hardware redundancy components such as lock-step are unavailable or infeasible for industry practitioners (due to their low maintainability levels [89]), virtualization-based methods can be used as a hybrid redundancy technique to ensure system availability [74,89–91]. For example, hypervisors, the most commonly used technologies for implementing virtualization, can leverage hardware security primitives (e.g., MMU and IOMMU) to establish redundant systems by either assuming a replica per virtual machine (VM) [74,89] or implementing nested virtualization and supporting multiple replicas in a single VM [90]. Since hypervisors have full control over hardware resources, they can ensure the simultaneous and isolated execution of various replicas while also preventing one VM from impacting the behavior of others. Additionally, since they are mostly implemented in software, they stand out against more hardware focused solutions in flexibility and maintainability levels. However, these advantages come with the price of significant latency and overhead, which can be justified with the maintenance of checkpoints in software rather than hardware [89].

4.1. Software, Hardware, and Hybrid Redundancy—Pros & Cons

Overall, each redundancy technique can be classified based on: (i) deployment approach, i.e., software (SW), hardware (HW), or both; (ii) area, referring to the amount of required resources; (iii) performance overhead, refers to the performance penalty inserted into the system; (iv) flexibility, which assesses the scalability and adaptability of the technique; (v) maintainability, focusing on whether the system can be updated, recovered, or diagnosed easily; and (vi) fault coverage, evaluating the system's resistance to different types of faults. Table 5 provides an overview of several redundancy techniques and their classifications.

Table 5. Comparison between different redundancy techniques. ✓—Supported approach.

Redundancy Technique	Approach		Area	Performance Overhead	Flexibility	Maintainability	Fault Coverage
	SW	HW					
Single-version	✓		$(1 \Rightarrow N)x$	High	High	High	Low
N-version programming	✓		$(1 \Rightarrow N)x$	High	High	High	Medium
Time-based redundancy	✓		$(1 \Rightarrow N)x$	High	High	High	Medium
Parity check	✓		-	-	High	High	Low
Cyclic code checksum	✓		-	-	High	High	Low
TMR		✓	$\sim 3x$	Low	Low	Low	High
DWC		✓	$\sim 2x$	Medium	Low	Low	High
NMR		✓	$(1 \Rightarrow N)x$	Low	Medium	Low	High
DCLS	✓	✓	$\sim 2x$	Medium	Medium	Medium	High
TCLS	✓	✓	$\sim 3x$	Low	Medium	Medium	High
Hypervisors	✓	✓	$(1 \Rightarrow N)$	Medium	High	High	High

Software-based redundant systems require a large amount of memory to create one-to-N replicas of these techniques and assume lower performance in terms of execution time when compared to hardware redundancy. As some advantages facing HW-based techniques, these techniques also offer a more flexible, less expensive, and easier-to-maintain implementation. From these techniques, time-based and N-version programming techniques stand out due to their diversity in implementation, either in multi-version or mixing between software and time redundancy capabilities. Information redundancy techniques like parity check and cyclic code checksum rely on additional computation bits instead of replicating system resources and, for this reason, they could not be fairly compared to the rest of the techniques in terms of area or performance. Regarding hardware-based techniques, they offer faster fault detection with smaller execution time overhead and high fault coverage. However, these benefits come at the cost of reduced flexibility and maintainability, as well as increased size, weight, power, and cost (SWPa-C) metrics compared to software-based redundancy options. Among these techniques, TMR demonstrates higher performance levels than DMR due to the capability to hide faults provided by the voting system. NMR offers more flexibility than other hardware redundancy techniques due to support for multiple replica configurations. Hybrid redundancy techniques can combine various redundancy types (e.g., mixing hardware and software redundancy), providing high flexibility and fault coverage levels. Similar to TMR, the TCLS technique also delivers high-performance levels due to the voting system. Lastly, high maintainability and flexibility levels can be achieved through the software-based nature of hypervisor hybrid

solutions. However, it comes with the costs of performance overhead. A use case example of how to use virtualization-based technique to implement redundancy is presented in following Section 4.2.

4.2. Use Case Example

In recent years, automotive system features have been substantially increasing [2], demanding more complex ECUs. Such features go from non-critical functionalities categorized in quality management (QM) levels (e.g., infotainment system) to systems classified with ASIL-D (e.g., SBW systems) [2]. The co-existence of several subsystems with different purposes and criticality levels in the same platforms, also known as mixed-criticality Systems (MCS), creates a challenge of minimizing the SWPa-C metrics while maintaining the same levels of performance and dependability. To address these challenges, recent studies demonstrate how virtualization-based technologies, particularly the use of static partitioning hypervisors (SPHs), could effectively consolidate MCSs onto the same hardware while attending to security and safety concerns [92].

A practical fault-tolerance approach to implement redundancy and address the aforementioned challenges is through virtualization-based techniques, i.e., by adhering to SPH design and creating multiple replicas using VMs. Each VM is a replica assigned to a single physical CPU (pCPU). As a result, this approach typically features as many replicas as the number of available pCPUs. However, by statically assigning resources among replicas and considering the limited resources available on automotive platforms, the number of subsystems that can run on the same platform is restricted, significantly limiting the system's flexibility. One potential solution is to increase the number of VM per pCPU by relying on nested virtualization and implementing a VM-scheduling mechanism in the hypervisor, such as using VM-stacking [90]. With the VM-stacking mechanism, the hypervisor can expand the number of VMs a pCPU can handle (Child 0 and Child 1 on Figure 10). This method enables the simultaneous and independent execution of multiple replicas while providing flexibility properties with numerous nested VMs with diverse criticality levels.

Figure 10 demonstrates a possible system configuration using SPH and VM stacking to implement a fault-tolerant system. This design follows a TMR setup, requiring the platform to have a minimum of three pCPUs with each replica requiring a dedicated nested VM known as Monitor (i) to implement the VM stack mechanism, (ii) to manage a voting mechanism, and (iii) to ensure synchronization between multiple replicas.

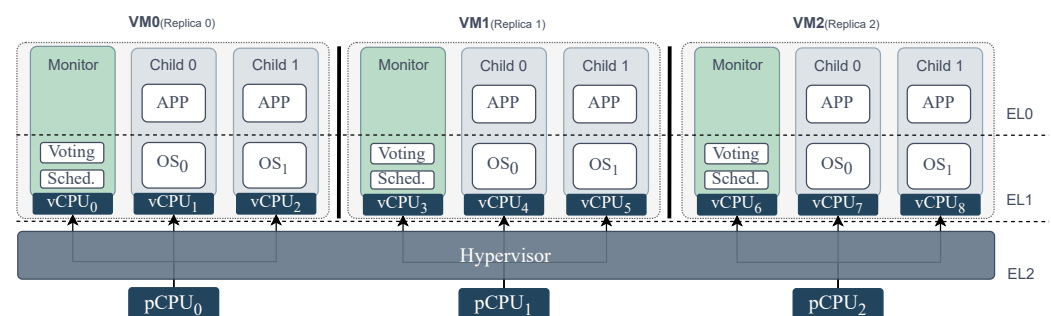


Figure 10. Virtualization-based redundancy system leveraging SPH combined with VM-stacking features.

VM-Stacking Mechanism. To provide scalability and support multiple VMs with different criticality levels, the Monitor schedules the processor execution between each child. Each child runs on a separate virtual CPU (vCPU) as a nested VM. The Monitor utilizes a

VM-stack and follows the last-in-first-out (LIFO) principle to coordinate the execution of each child.

Voting mechanism. To identify the presence faults or errors, a voting mechanism must compare all replicas at the monitor level through a loosely coupled technique, where the voting process is triggered when each child executes a critical operation. For example, this process can be achieved through para-virtualization by using hypercalls, i.e., as soon as child executes a critical process, it also triggers a hypercall for further detection procedures. Once the execution reaches the Monitor, the voting mechanism should check for faults or errors in all replicas, potentially through shared memory between VMs. In the event of a fault, it should also identify the faulty replica, relying, for example, on the checksum computation. After this process, the execution should return to SPH (EL2 privileged level) and communicate the result of the voting process, possibly through a dedicated flag. This flag is then used to control the execution flow, determining whether the system should initiate a checkpoint or a recovery process.

Checkpoint and recovery mechanism. When the execution reaches the hypervisor level, the SPH should perform different actions based on the voting decision. For example, in an agreement case between all replicas with a roll-back recovery system involved, the execution should initiate a checkpoint procedure by keeping the system state safe (by saving the correspondent processor registers). Conversely, if a faulty replica is detected, a recovery mechanism should restore the child's state to the last safe checkpoint. If all replicas vote differently, the SPH could also utilize the checkpoint state to revert them to the most recent safe and agreement state.

Synchronization mechanism. The lack of synchronization may cause the system to be compared at different times during execution, creating permanent faults and a loss of system availability. Therefore, addressing synchronization challenges and ensuring the comparison of replicas at the correct execution time is also an essential process. A possible synchronization point could be placed after the execution of critical functionalities and before the voting process (i.e., at the monitor level), preventing comparisons of the replicas at different execution times. As a result, once a processor from any of the replicas reaches this synchronization point, it will wait for the other replicas and initiate a timer with a predetermined overflow value. When all replicas reach the synchronization point, the timer is stopped, and the system moves to the voting process safely. If one replica takes too long to reach the synchronization point, the timer triggers an interrupt, which (i) indicates the presence of a faulty replica and (ii) initiates a recovery process. This recovery process involves a voting comparison between the two remaining *children* to evaluate if only one faulty replica exists. Depending on the result, the system recovers the faulty child or the entire system. If both childs match, the system forces the desynchronized child to recover. Conversely, if *children* do not match, the system can enter a failure state, which forces all *children* to return to a previously healthy state.

Dependability features in SPHs. For redundant strategies depending on SPH, some essential dependability features to ensure the system's availability must be included. Firstly, the SPH needs to handle hypercalls (i.e., detection requests) from its *children* and generate an identification value to represent its state. This is crucial for recovering the child's state in the event of a fault or error. Secondly, given that the SPH has access to all system resources, it should be responsible for processing the checkpoint and recovery state. Thirdly, as multiple replicas may request access to the same device, SPHs should regulate these accesses securely, possibly through trap-and-emulate strategies. Lastly, the SPH should manage interrupt handling by redirecting device interrupts to the respective child of multiple replicas.

Determinism and real-time constraints. As previously mentioned, virtualization can adversely affect system performance, primarily due to the overhead introduced by two-stage memory address translation. This impact stems from the additional latency incurred during translation lookaside buffer (TLB) misses and page table walks, which can introduce non-determinism due to factors like bus contention. However, Martins et al. [92] demonstrated that the performance impact of shadow paging is minimal (less than 1%) when using 2 MB superpages, as the larger page size reduces pressure on the TLB. Despite these optimizations, the fundamental challenge of non-deterministic latency remains in systems with virtualized memory. To address this, emerging approaches avoid memory virtualization altogether by maintaining a one-to-one mapping between virtual and physical addresses. In such configurations, the hypervisor still enforces spatial isolation but relies on hardware primitives, such as ARM's two-level memory protection unit (MPU) or RISC-V's supervisor-mode physical memory protection (SPMP), to provide isolation [93,94]. This method preserves the isolation guarantees typically provided by virtualization while eliminating the timing unpredictability associated with dynamic address translation, thereby enabling more predictable execution for real-time automotive applications.

4.3. Currently Used Redundancy

Upon reviewing the automotive portfolios of leading manufacturers, it is evident that several products are designed specifically for safety-critical applications. Table 6 highlights the existing support by MCU manufacturers for automotive systems with lockstep technology. STMicroelectronics' Stellar G, P, and E platforms incorporate Arm Cortex R52+/-M4 and -M7 cores that offer lockstep configurations ranging from single to multi-core lockstep (MCLS). These platforms are designed for real-time and safety performance purposes (with ASIL-D levels). Additionally, Cortex-R52+ introduce hardware virtualization-based security primitives, i.e., level 2 MPU, which facilitates hypervisor implementations and consequently redundant hybrid solutions like the one presented in Section 4.2. Some of Hercules platforms from Texas Instruments, certified to ISO26262 [23] and IEC61508 [95], are equipped with ARM Cortex-R4F/R5F processors that offer DCLS configurations. Infineon's Aurix platforms, with TC2x/3x/4x cores, comply with ASIL-D standards effortlessly with TCLS configurations. Renesas' RH850/U2A16 MCUs feature DCLS configuration while providing high-performance levels with low power consumption. NXP's MPC564xL and S32E/Z/K platforms, equipped with E200 Z4 and Arm Cortex-R52/-M33/-M7 cores offer DCLS configurations and address the ISO 26262 safety standard up to ASIL-D requirements.

Fault-tolerance methods are the key to achieving the availability pillar mentioned in the security triad, and there's a clear industry trend towards the adoption of lockstep technologies. As demonstrated, the use of lockstep technology proliferates across all automotive MCU companies, being available in different forms (e.g., DCLS, TCLS, and MCLS). However, it is worth noting that other technologies, such as hardware virtualization-based technologies found in recent cores like the Arm Cortex-R52, are also being integrated to enhance system safety and security. This level of technology allows MCS consolidation with different ASIL levels onto the same platform, making it a strong candidate to complement or substitute lockstep solutions with the form of SPHs, as discussed in Section 4.2.

Hardware support for real-time virtualization. The use of virtualization heavily depends on the availability of supporting hardware primitives, as full software-based virtualization incurs significant performance penalties. Unfortunately, such hardware support is typically limited to high-end processors, such as the MAC57D5xx family from NXP, which represent only a small fraction of the chips currently deployed in vehicles. Moreover, implementing memory virtualization through an MMU reduces the determinism of the system, which is undesirable given the constraints of most automotive applications. How-

ever, with the recent adoption of the Arm Cortex-R52 in the automotive industry, new virtualization-capable hardware primitives are starting to appear in modern ECUs, while enabling systems to maintain determinism. Nonetheless, currently, only the Stellar family from STMicroelectronics offers a viable platform for deploying the proposed use case, as it supports a two-level MPU. These limitations restrict the integration of mixed-criticality systems onto a single platform, where latency-sensitive and high-integrity sensor processing pipelines must co-exist. Without strong hardware isolation and low-overhead virtualization support, it becomes infeasible to consolidate high- and low-ASIL workloads in compliance with ISO 26262 constraints.

Table 6. Top 5 automotive semiconductor vendors' lockstep support. ✓—Primitive supported.

Vendor	Platform Name	Lockstep Type	CPU Architecture	Level 2 MPU
Infineon Technologies	Aurix	TCLS	TC2x/TC3x/TC4x	
NXP Semiconductors	MPC564xL/S32E/Z/K	DCLS	Arm Cortex R52/Cortex-M33/E200 Z4/Cortex-M7	
Renesas Electronics	RH850/U2A16	DCLS	RH850 G4MH	
Texas Instruments Inc.	Hercules	DCLS	Arm Cortex-R4F/Arm Cortex-R5F	
STMicroelectronics	Stellar	MCLS	Arm Cortex R52+/Cortex-M4/Cortex-M7	✓

5. Final Discussion and Future Directions

Currently, modern automotive systems can contain over 70 ECUs [96]. With the advancements in connectivity, such as vehicle-to-everything (V2X) communication, along with the advancement of ADAS and autonomous driving features, the number of ECUs per vehicle is predicted to grow. As a result, vehicles are more susceptible to cyber-attacks, making it crucial to explore several critical areas to enhance vehicle safety and security. The following key areas are essential for future research and industry developments.

- **Addressing hardware security primitives:** The majority of automotive solutions do not meet all the requirements for memory and I/O isolation as they lack robust memory protection and system-wide protection technologies. Despite the recent shift towards more mature security-focused MCUs, this transition is still in its early stages. Moreover, security considerations must be integrated from the ground up when developing new vehicular architectures, going all the way from the simplest ECUs to complex inter-vehicular communication networks. As security threats emerge continuously and as vehicles are deployed in the field, possibly for decades, it is necessary to secure them with more mature security-focused hardware architectures, eliminating the compromise between performance and security.
- **Addressing reliability challenges:** Automotive ECUs currently incorporate fault-tolerant components to achieve high-reliability levels, primarily through redundancy. Nevertheless, these systems face challenges in meeting the strict safety and dependability requirements of automotive systems without compromising hard real-time

constraints and increasing costs. The additional redundancy in fault-tolerant designs must not compromise the real-time performance of automotive systems. As such, as ECUs start to become more complex, the need for more lightweight and hybrid redundancy solutions is growing.

- **Addressing system integration and partitioning:** Integrating different ASIL systems into the same ECU is essential to address current SWAP-C metrics for automotive systems. To overcome these challenges, future research should explore advanced partitioning techniques to isolate safety-critical functions from non-critical ones onto the same platform. Virtualization-based solutions can help to consolidate MCSs while maintaining the robust security and reliability of each independent system.
- **Enhancing protection against side-channel attacks:** Side-channel attacks pose significant risks to automotive ECUs. From electromagnetic radiation to timing variations, attackers can exploit these side-channels to gain insight into system operations and potentially compromise sensitive data. However, side-channel attack can be challenging to defend against, as they may occur due to implementation flaws rather than design mistakes. Therefore, to effectively protect against side-channel attack, a thorough understanding of all the components within an ECU is essential. Future research should concentrate on actively detecting potential side-channels and developing advanced techniques to detect and mitigate these attacks.

By addressing these areas, both research and industry can make significant strides toward developing safer, more reliable, and secure automotive systems. Continued innovation and collaboration will be key to meeting the evolving demands of the automotive landscape. Additionally, vehicle safety and security mechanisms are often designed independently, which may lead to incompatible measures. To address this, seamless integration between safety and security protocols is essential for manufacturers to achieve higher levels of both. Therefore, future research should focus on examining the interrelations between these measures, as true safety cannot be achieved without robust security.

6. Conclusions

As the automotive sector advances toward full autonomy and pervasive connectivity, the security and reliability of embedded systems emerge as critical design requirements. This paper examined the architectural evolution of electronic control units, highlighting the security challenges introduced by growing system complexity and interconnectivity. Existing designs often lack resilience against advanced cyber threats, particularly side-channel attacks, due to computational constraints and the absence of robust hardware-level security primitives. In parallel, the increasing dependence on sensor data for critical decision making reinforces the need for memory and system-wide isolation mechanisms to be integrated directly into hardware platforms. To support high system availability and fault tolerance, the analysis explored redundancy techniques spanning software, hardware, and hybrid approaches. Virtualization-based solutions were identified as a promising means of consolidating mixed-criticality systems while maintaining safety, performance, and maintainability.

This work offers a cross-layer architectural perspective that brings together fault-tolerance mechanisms, hardware security primitives, and side-channel threat modeling under a common framework. It outlines key design considerations for deploying secure and reliable embedded systems in autonomous vehicles, emphasizing both technical feasibility and practical constraints. By grounding the analysis in commercial hardware platforms and realistic use cases, the study provides concrete guidance for system designers addressing safety and security requirements in next-generation automotive systems. Treating security

and safety as inseparable pillars is essential for building resilient and trustworthy platforms capable of supporting the next generation of autonomous vehicles.

Author Contributions: Conceptualization, S.P., J.A. and T.G.; methodology, S.P. and T.G.; validation, L.C., J.S., S.P. and T.G.; investigation, L.C., J.S., T.G. and S.P.; resources, S.P. and T.G.; writing—original draft preparation, L.C., J.S., T.G. and S.P.; writing—review and editing, L.C., J.S., T.G. and S.P.; supervision, S.P. and T.G.; project administration, S.P., J.A. and T.G.; funding acquisition, S.P., J.A. and T.G. All authors have read and agreed to the published version of the manuscript.

Funding: This work is supported by national funds, through the Operational Competitiveness and Internationalization Programme (COMPETE 2020) [Project n° 179491; Funding Reference: SIFN-01-9999-FN-179491].

Data Availability Statement: Data are contained within the article.

Conflicts of Interest: Author José Azevedo was employed by the company Bosch Car Multimedia. The remaining authors declare that the research was conducted in the absence of any commercial or financial relationships that could be construed as a potential conflict of interest

References

1. Daily, M.; Medasani, S.; Behringer, R.; Trivedi, M. Self-Driving Cars. *Computer* **2017**, *50*, 18–23. [CrossRef]
2. Staron, M. *Automotive Software Architectures*; Springer: Berlin/Heidelberg, Germany, 2021.
3. Urmson, C.; Anhalt, J.; Bagnell, D.; Baker, C.; Bittner, R.; Clark, M.N.; Dolan, J.; Duggins, D.; Galatali, T.; Geyer, C.; et al. Autonomous driving in urban environments: Boss and the Urban Challenge. *J. Field Robot.* **2008**, *25*, 425–466. [CrossRef]
4. Shahian Jahromi, B.; Tulabandhula, T.; Cetin, S. Real-Time Hybrid Multi-Sensor Fusion Framework for Perception in Autonomous Vehicles. *Sensors* **2019**, *19*, 4357. [CrossRef]
5. Wilwert, C.; Navet, N.; Song, Y.Q.; Simonot-Lion, F. Design of automotive X-by-Wire systems. In *The Industrial Communication Technology Handbook*; Zurawski, R., Ed.; CRC Press: Boca Raton, FL, USA, 2005.
6. Heiser, G. No Safety Without (Cyber-)Security! *ACM SIGBED* 2020. Available online : <https://microkerneldude.org/2020/11/07/no-safety-without-cyber-security/> (accessed on 20 August 2024).
7. Miller, C. Lessons learned from hacking a car. *IEEE Design Test* **2019**, *36*, 7–9. [CrossRef]
8. Greenberg, A. The Jeep Hackers Are Back to Prove Car Hacking Can Get Much Worse. *WIRED* 2016. Available online: <https://www.wired.com/2016/08/jeep-hackers-return-high-speed-steering-acceleration-hacks/> (accessed on 23 April 2024).
9. Greenberg, A. This Bluetooth Attack Can Steal a Tesla Model X in Minutes. *WIRED* 2020. Available online: <https://www.wired.com/story/tesla-model-x-hack-bluetooth/> (accessed on 23 April 2024).
10. Gomes, T.; Sousa, P.; Silva, M.; Ekpanyapong, M.; Pinto, S. FAC-V: An FPGA-Based AES Coprocessor for RISC-V. *J. Low Power Electron. Appl.* **2022**, *12*, 50. [CrossRef]
11. Kocher, P.C. Timing Attacks on Implementations of Diffie-Hellman, RSA, DSS, and Other Systems. In *Proceedings of the Advances in Cryptology—CRYPTO’96*, Santa Barbara, CA, USA, 18–22 August 1996; Kobitz, N., Ed.; Springer: Berlin/Heidelberg, Germany, 1996; pp. 104–113.
12. Kocher, P.; Horn, J.; Fogh, A.; Genkin, D.; Gruss, D.; Haas, W.; Hamburg, M.; Lipp, M.; Mangard, S.; Prescher, T.; et al. Spectre Attacks: Exploiting Speculative Execution. In *Proceedings of the IEEE Symposium on Security and Privacy (SP)*, San Francisco, CA, USA, 19–23 May 2019; pp. 1–19.
13. Lipp, M.; Schwarz, M.; Gruss, D.; Prescher, T.; Haas, W.; Fogh, A.; Horn, J.; Mangard, S.; Kocher, P.; Genkin, D.; et al. Meltdown: Reading Kernel Memory from User Space. In *Proceedings of the 27th USENIX Security Symposium (USENIX Security 18)*, Baltimore, MD, USA, 15–17 August 2018; pp. 973–990.
14. Luo, M.; Myers, A.C.; Suh, G.E. Stealthy Tracking of Autonomous Vehicles with Cache Side Channels. In *Proceedings of the 29th USENIX Security Symposium (USENIX Security 20)*, Boston, MA, USA, 12–14 August 2020; pp. 859–876.
15. Trippel, T.; Weisse, O.; Xu, W.; Honeyman, P.; Fu, K. WALNUT: Waging Doubt on the Integrity of MEMS Accelerometers with Acoustic Injection Attacks. In *Proceedings of the 2017 IEEE European Symposium on Security and Privacy (EuroS&P)*, Paris, France, 26–28 April 2017; pp. 3–18.
16. Wirthlin, M.J.; Keller, A.M.; McCloskey, C.; Ridd, P.; Lee, D.; Draper, J. SEU Mitigation and Validation of the LEON3 Soft Processor Using Triple Modular Redundancy for Space Processing. In *Proceedings of the 2016 ACM/SIGDA International Symposium on Field-Programmable Gate Arrays*, Monterey, CA, USA, 21–23 February 2016; pp. 205–214.
17. Koopman, P.; Wagner, M. Autonomous Vehicle Safety: An Interdisciplinary Challenge. *IEEE Intell. Transp. Syst. Mag.* **2017**, *9*, 90–96. [CrossRef]

18. Pinto, S.; Tavares, A.; Montenegro, S. Space and time partitioning with hardware support for space applications. In Proceedings of the DATA Systems in Aerospace Conference, DASIA, Tallinn, Estonia, 10–12 May 2016.
19. Masmano, M.; Ripoll, I.; Crespo, A.; Jean-Jacques, M. XtratuM: A Hypervisor for Safety Critical Embedded Systems. In Proceedings of the 11th Real Time Linux Workshop, Dresden, Germany, 28–30 September 2009.
20. Placek, M. Automotive Electronics Worldwide—Statistics & Facts. 2023. Available online: : <https://www.statista.com/topics/7983/automotive-electronics-worldwide/> (accessed on 12 April 2024).
21. Schäuffele, J.; Zurawka, T. *Automotive Software Engineering: Principles, Processes, Methods, and Tools*; SAE International: Warrendale, PA, USA, 2005.
22. Eckstein, L.; Hesse, L.; Klein, M. Steer-by-Wire, Potential, and Challenges. In *Encyclopedia of Automotive Engineering*; John Wiley & Sons, Ltd.: Hoboken, NJ, USA, 2014; pp. 1–14.
23. ISO 26262-1:2018; Road Vehicles—Functional Safety—Part 1: Vocabulary. International Organization for Standardization: Geneva, Switzerland, 2018.
24. Bosch Mobility. Vehicle Control Unit. Available online: <https://www.bosch-mobility.com/en/solutions/control-units/vehicle-control-unit/> (accessed on 24 March 2025).
25. McKay, B.; Qian, L. Integrated Power Electronics: The Road Merges. Technical Report, APTIV. 2022. Available online: <https://www.aptiv.com/en/insights/article/integrated-power-electronics-the-road-merges> (accessed on 24 March 2024).
26. Bosch Mobility. The Electric Power Steering System with Belt Drive Servo Unit. Available online: <https://www.bosch-mobility.com/en/solutions/steering/electric-power-steering-belt-drive-servo-unit/> (accessed on 24 March 2025).
27. Hitachi Automotive Systems Americas, Inc. Steering System. Available online: <https://www.hitachi-automotive.us/Products/oem/DCS/Steering/index.htm> (accessed on 24 March 2025).
28. Infineon Technologies AG. Airbag Control. Available online: <https://www.infineon.com/cms/en/applications/automotive/chassis-safety-and-adas/airbag-system/> (accessed on 24 March 2025).
29. Sukumar, K.K.; Oberkönig, M.; Noopuran, S.; Cypress. Functional Safety for Automotive HMI Systems. 2019. Available online: <https://passive-components.eu/functional-safety-for-automotive-hmi-systems/> (accessed on 24 March 2025).
30. Liu, T.; Liu, J.; Wang, J.; Zhang, H.; Zhang, B.; Ma, Y.; Sun, M.; Lv, Z.; Xu, G. Pseudolites to Support Location Services in Smart Cities: Review and Prospects. *Smart Cities* **2023**, *6*, 2081–2105. [CrossRef]
31. Roriz, R.; Cabral, J.; Gomes, T. Automotive LiDAR Technology: A Survey. *IEEE Trans. Intell. Transp. Syst.* **2022**, *23*, 6282–6297. [CrossRef]
32. Joubert, N.; Reid, T.G.R.; Noble, F. Developments in Modern GNSS and Its Impact on Autonomous Vehicle Architectures. In Proceedings of the 2020 IEEE Intelligent Vehicles Symposium (IV), Las Vegas, NV, USA, 19 October–13 November 2020; pp. 2029–2036.
33. Costa, D.; Cuomo, L.; Oliveira, D.; Savino, I.M.; Morelli, B.; Martins, J.; Biasci, A.; Pinto, S. IRQ Coloring and the Subtle Art of Mitigating Interrupt-generated Interference. *arXiv* **2023**, arXiv:22308.00997.
34. Costa, D.; Moreira, G.; Oliveira, A.; Martins, J.; Pinto, S. SP-IMPact: A Framework for Static Partitioning Interference Mitigation and Performance Analysis. In Proceedings of the Sixth Workshop on Next Generation Real-Time Embedded Systems (NG-RES 2025), Barcelona, Spain, 20–22 January 2025; Volume 128, pp. 5:1–5:15.
35. Zalman, R. *Rugged Autonomous Vehicles*; Morgan Kaufmann: Burlington, MA, USA, 2017.
36. Vailshery, L.S.. Microcontrollers—Statistics & Facts. 2024. Available online: <https://www.statista.com/topics/9939/microcontrollers/#topicOverview> (accessed on 12 April 2024).
37. Pinto, S.; Santos, N. Demystifying Arm TrustZone: A Comprehensive Survey. *ACM Comput. Surv.* **2019**, *51*, 1–36. [CrossRef]
38. PSA Certified. What Is a Root of Trust? Available online: <https://www.psacertified.org/blog/what-is-a-root-of-trust/> (accessed on 24 March 2025).
39. McGrath, T.; Bagci, I.E.; Wang, Z.M.; Roedig, U.; Young, R.J. A PUF taxonomy. *Appl. Phys. Rev.* **2019**, *6*, 011303. [CrossRef]
40. Wolf, M.; Gendrullis, T. Design, implementation, and evaluation of a vehicular hardware security module. In Proceedings of the 14th International Conference on Information Security and Cryptology, Seoul, Republic of Korea, 30 November–2 December 2011; pp. 302–318.
41. Goodin, D. Found: New Android Malware with Never-Before-Seen Spying Capabilities. 2018. Available online: <https://arstechnica.com/information-technology/2018/01/found-new-android-malware-with-never-before-seen-spying-capabilities/> (accessed on 24 March 2025).
42. Dave, L. Apple’s App Store Infected with XcodeGhost Malware in China. 2015. Available online: <https://www.bbc.com/news/technology-34311203> (accessed on 24 March 2025).
43. Cerdeira, D.; Santos, N.; Fonseca, P.; Pinto, S. SoK: Understanding the Prevailing Security Vulnerabilities in TrustZone-assisted TEE Systems. In Proceedings of the 2020 IEEE Symposium on Security and Privacy (SP), San Francisco, CA, USA, 18–21 May 2020; pp. 1416–1432.

44. Hu, S.; Chen, Q.A.; Joung, J.; Carlak, C.; Feng, Y.; Mao, Z.M.; Liu, H.X. CVShield: Guarding Sensor Data in Connected Vehicle with Trusted Execution Environment. In Proceedings of the Second ACM Workshop on Automotive and Aerial Vehicle Security, New Orleans, LA, USA, 18 March 2020; Association for Computing Machinery: New York, NY, USA, 2020.
45. Pinto, S.; Araujo, H.; Oliveira, D.; Martins, J.; Tavares, A. Virtualization on TrustZone-Enabled Microcontrollers? Voilà! In Proceedings of the 2019 IEEE Real-Time and Embedded Technology and Applications Symposium (RTAS), Montreal, QC, Canada, 16–18 April 2019 ; pp. 293–304.
46. Oliveira, D.; Gomes, T.; Pinto, S. uTango: An Open-Source TEE for IoT Devices. *IEEE Access* **2022**, *10*, 23913–23930. [[CrossRef](#)]
47. Garlati, C.; Pinto, S. A Clean Slate Approach to Linux Security RISC-V Enclaves. In Proceedings of the Embedded World Conference (EWC), Nuremberg, Germany, 25–27 February 2020 .
48. Rivain, M.; Prouff, E. Provably Secure Higher-Order Masking of AES. In Proceedings of the Cryptographic Hardware and Embedded Systems, CHES 2010, Santa Barbara, CA, USA, 17–20 August 2010 ; pp. 413–427.
49. Yan, C.; Shin, H.; Bolton, C.; Xu, W.; Kim, Y.; Fu, K. SoK: A Minimalist Approach to Formalizing Analog Sensor Security. In Proceedings of the 2020 IEEE Symposium on Security and Privacy (SP), San Francisco, CA, USA, 18–21 May 2020 ; pp. 233–248.
50. Petit, J.; Stottelaar, B.; Feiri, M. Remote Attacks on Automated Vehicles Sensors: Experiments on Camera and LiDAR. In Proceedings of the Black Hat Europe, Amsterdam, The Netherlands, 10–13 November 2015 .
51. Shin, H.; Kim, D.; Kwon, Y.; Kim, Y. Illusion and Dazzle: Adversarial Optical Channel Exploits Against Lidars for Automotive Applications. In Proceedings of the Cryptographic Hardware and Embedded Systems—CHES 2017, Taipei, Taiwan, 25–28 September 2017.
52. Yan, C.; Xu, W.; Liu, J. Can You Trust Autonomous Vehicles: Contactless Attacks against Sensors of Self-driving Vehicle. In Proceedings of the DEF CON 24, Las Vegas, NV, USA, 4–7 August 2016 .
53. Camurati, G.; Poeplau, S.; Muench, M.; Hayes, T.; Francillon, A. Screaming Channels: When Electromagnetic Side Channels Meet Radio Transceivers. In Proceedings of the 2018 ACM SIGSAC Conference on Computer and Communications Security, Toronto, ON, Canada, 15–19 October 2018; Association for Computing Machinery: New York, NY, USA, 2018 .
54. Zhan, Z.; Zhang, Z.; Liang, S.; Yao, F.; Koutsoukos, X. Graphics Peeping Unit: Exploiting EM Side-Channel Information of GPUs to Eavesdrop on Your Neighbors. In Proceedings of the 2022 IEEE Symposium on Security and Privacy (SP), San Francisco, CA, USA, 22–26 May 2022.
55. Gandolfi, K.; Mourtel, C.; Olivier, F. Electromagnetic Analysis: Concrete Results. In Proceedings of the Third International Workshop on Cryptographic Hardware and Embedded Systems, Paris, France, 14–16 May 2001 ; pp. 251–261.
56. Xu, W.; Yan, C.; Jia, W.; Ji, X.; Liu, J. Analyzing and Enhancing the Security of Ultrasonic Sensors for Autonomous Vehicles. *IEEE Internet Things J.* **2018**, *5*, 5015–5029. [[CrossRef](#)]
57. Bhunia, S.; Tehranipoor, M. Side-Channel Attacks. In *Hardware Security*; Bhunia, S., Tehranipoor, M., Eds.; Morgan Kaufmann: Cambridge, MA, USA, 2019.
58. Van Bulck, J.; Piessens, F.; Strackx, R. Nemesis: Studying Microarchitectural Timing Leaks in Rudimentary CPU Interrupt Logic. In Proceedings of the 2018 ACM SIGSAC Conference on Computer and Communications Security, Toronto, ON, Canada, 15–19 October 2018; pp. 178–195.
59. Rodrigues, C.; Oliveira, D.; Pinto, S. BUSted!!! Microarchitectural Side-Channel Attacks on the MCU Bus Interconnect. In Proceedings of the 2024 IEEE Symposium on Security and Privacy (SP), San Francisco, CA, USA, 20–23 May 2024 ; pp. 3679–3696.
60. Van Bulck, J.; Mühlberg, J.T.; Piessens, F. VulCAN: Efficient Component Authentication and Software Isolation for Automotive Control Networks. In Proceedings of the 33rd Annual Computer Security Applications Conference, Orlando, FL, USA, 4–8 December 2017; Association for Computing Machinery: New York, NY, USA, 2017; pp. 225–237.
61. Avizienis, A. Toward systematic design of fault-tolerant systems. *Computer* **1997**, *30*, 51–58. [[CrossRef](#)]
62. Pimentel, A.D. A Case for Security-Aware Design-Space Exploration of Embedded Systems. *J. Low Power Electron. Appl.* **2020**, *10*, 22. [[CrossRef](#)]
63. Akter, S.; Khalil, K.; Bayoumi, M. A Survey on Hardware Security: Current Trends and Challenges. *IEEE Access* **2023**, *11*, 77543–77565. [[CrossRef](#)]
64. Avizienis, A.; Laprie, J.C.; Randell, B.; Landwehr, C. Basic concepts and taxonomy of dependable and secure computing. *IEEE Trans. Dependable Secur. Comput.* **2004**, *1*, 11–33. [[CrossRef](#)]
65. Richter-Brockmann, J.; Sasdrich, P.; Bache, F.; Güneysu, T. Concurrent error detection revisited: Hardware protection against fault and side-channel attacks. In Proceedings of the 15th International Conference on Availability, Reliability and Security, Virtual Event, 25–28 August 2020 .
66. Wang, L.; Hu, M.; Bian, Y.; Guo, G.; Li, S.E.; Chen, B.; Zhong, Z. Periodic Event-Triggered Fault Detection for Safe Platooning Control of Intelligent and Connected Vehicles. *IEEE Trans. Veh. Technol.* **2024**, *73*, 5064–5077. [[CrossRef](#)]
67. Hasan, M.; Goraya, M.S. Fault tolerance in cloud computing environment: A systematic survey. *Comput. Ind.* **2018**, *99*, 156–172. [[CrossRef](#)]

68. Rosenstatter, T.; Strandberg, K.; Jolak, R.; Scandariato, R.; Olovsson, T. REMIND: A Framework for the Resilient Design of Automotive Systems. In Proceedings of the 2020 IEEE Secure Development (SecDev), Virtual Event, 28–30 September 2020; pp. 81–95.
69. Kasap, S.; Weber Wachter, E.; Zhai, X.; Ehsan, S.; McDonald-Maier, K. Novel lockstep-based fault mitigation approach for SoCs with roll-back and roll-forward recovery. *Microelectron. Reliab.* **2021**, *124*, 114297. [[CrossRef](#)]
70. Somani, A.K.; Vaidya, N.H. Understanding Fault Tolerance and Reliability. *Computer* **1997**, *30*, 45–50. [[CrossRef](#)]
71. Răzvan, S.; Csaba, S. Software Redundancy Implementation Strategy in Reconfigurable Hardware Framework. In Proceedings of the 2019 8th International Conference on Modern Power Systems (MPS), Cluj-Napoca, Romania, 21–23 May 2019; pp. 1–6.
72. Afzaal, U.; Lee, J.A. Low-cost Hardware Redundancy for Fault-mitigation in Power-constrained IoT Systems. In Proceedings of the 2020 International Conference on Information and Communication Technology Convergence (ICTC), Jeju Island, Republic of Korea, 21–23 October 2020; pp. 60–62.
73. Mallavarapu, P.; Upadhyay, H.N.; Rajkumar, G.; Elamaram, V. Fault-tolerant digital filters on FPGA using hardware redundancy techniques. In Proceedings of the 2017 International Conference of Electronics, Communication and Aerospace Technology (ICECA), Coimbatore, India, 20–22 April 2017; Volume 2, pp. 256–259.
74. Alavizadeh, H.; Hong, J.B.; Kim, D.S.; Jang-Jaccard, J. Evaluating the effectiveness of shuffle and redundancy MTD techniques in the cloud. *Comput. Secur.* **2021**, *102*, 102091. [[CrossRef](#)]
75. Afonso, F.; Silva, C.; Brito, N.; Montenegro, S.; Tavares, A. Aspect-oriented fault tolerance for real-time embedded systems. In Proceedings of the 2008 AOSD Workshop on Aspects, Components, and Patterns for Infrastructure Software, Brussels, Belgium, 1 April 2008.
76. Shajahan, M.A. Fault Tolerance and Reliability in AUTOSAR Stack Development: Redundancy and Error Handling Strategies. *Technol. Manag. Rev.* **2018**, *3*, 27–45.
77. Ertugrul, E.; Sahingoz, O.K. Fault Tolerance in Real-Time Systems: A Review. In Proceedings of the Intelligent Systems Design and Applications, Vellore, India, 6–8 December 2018; Abraham, A., Muhuri, P.K., Muda, A.K., Gandhi, N., Eds.; Springer International Publishing: Berlin/Heidelberg, Germany, 2018; pp. 283–293.
78. Abdulnabi, M.S.; Ahmed, H. Design of Efficient Cyclic Redundancy Check-32 using FPGA. In Proceedings of the 2018 International Conference on Computer, Control, Electrical, and Electronics Engineering (ICCCEEE), Khartoum, Sudan, 12–14 August 2018; pp. 1–5.
79. Gao, Z.; Cecati, C.; Ding, S.X. A Survey of Fault Diagnosis and Fault-Tolerant Techniques—Part I: Fault Diagnosis with Model-Based and Signal-Based Approaches. *IEEE Trans. Ind. Electron.* **2015**, *62*, 3757–3767. [[CrossRef](#)]
80. Şinca, R.; Szász, C. Fault-tolerant digital systems development using triple modular redundancy. *Int. Rev. Appl. Sci. Eng. IRASE* **2017**, *8*, 3–7. [[CrossRef](#)]
81. Abbaspour, A.; Mokhtari, S.; Sargolzaei, A.; Yen, K.K. A Survey on Active Fault-Tolerant Control Systems. *Electronics* **2020**, *9*, 1513. [[CrossRef](#)]
82. Safari, S.; Ansari, M.; Khdr, H.; Gohari-Nazari, P.; Yari-Karin, S.; Yeganeh-Khaksar, A.; Hessabi, S.; Ejlali, A.; Henkel, J. A Survey of Fault-Tolerance Techniques for Embedded Systems From the Perspective of Power, Energy, and Thermal Issues. *IEEE Access* **2022**, *10*, 12229–12251. [[CrossRef](#)]
83. Peña-Fernández, M.; Serrano-Cases, A.; Lindoso, A.; Cuenca-Asensi, S.; Entrena, L.; Morilla, Y.; Martín-Holgado, P.; Martínez-Álvarez, A. Hybrid Lockstep Technique for Soft Error Mitigation. *IEEE Trans. Nucl. Sci.* **2022**, *69*, 1574–1581. [[CrossRef](#)]
84. Iturbe, X.; Venu, B.; Ozer, E.; Poupat, J.L.; Gimenez, G.; Zurek, H.U. The Arm Triple Core Lock-Step (TCLS) Processor. *ACM Trans. Comput. Syst.* **2019**, *36*, 3323917. [[CrossRef](#)]
85. Kaufman, L.; Bhide, S.; Johnson, B. Modeling of common-mode failures in digital embedded systems. In Proceedings of the Annual Reliability and Maintainability Symposium, Los Angeles, CA, USA, 24–27 January 2000; 2000 Proceedings; International Symposium on Product Quality and Integrity (Cat. No.00CH37055); pp. 350–357.
86. Sim, M.T.; Zhuang, Y. A Dual Lockstep Processor System-on-a-Chip for Fast Error Recovery in Safety-Critical Applications. In Proceedings of the IECON 2020 The 46th Annual Conference of the IEEE Industrial Electronics Society, Singapore, 18–21 October 2020; pp. 2231–2238.
87. Kottke, T.; Steininger, A. A Reconfigurable Generic Dual-Core Architecture. In Proceedings of the International Conference on Dependable Systems and Networks (DSN'06), Philadelphia, PA, USA, 25–28 June 2006; pp. 45–54.
88. Marques, I.; Rodrigues, C.; Tavares, A.; Pinto, S.; Gomes, T. Lock-V: A heterogeneous fault tolerance architecture based on Arm and RISC-V. *Microelectron. Reliab.* **2021**, *120*, 114120. [[CrossRef](#)]
89. Ren, S.; Zhang, Y.; Pan, L.; Xiao, Z. Phantasy: Low-Latency Virtualization-Based Fault Tolerance via Asynchronous Prefetching. *IEEE Trans. Comput.* **2019**, *68*, 225–238. [[CrossRef](#)]
90. Tkachov, V.; Hunko, M.; Volotka, V. Scenarios for Implementation of Nested Virtualization Technology in Task of Improving Cloud Firewall Fault Tolerance. In Proceedings of the 2019 IEEE International Scientific-Practical Conference Problems of Infocommunications, Science and Technology (PIC S&T), Kyiv, Ukraine, 8–11 October 2019; pp. 759–763.

91. Lorch, J.R.; Baumann, A.; Glendenning, L.; Meyer, D.; Warfield, A. Tardigrade: Leveraging Lightweight Virtual Machines to Easily and Efficiently Construct Fault-Tolerant Services. In Proceedings of the 12th USENIX Symposium on Networked Systems Design and Implementation (NSDI 15), Oakland, CA, USA, 4–6 May 2015.
92. Martins, J.; Pinto, S. Shedding Light on Static Partitioning Hypervisors for Arm-based Mixed-Criticality Systems. In Proceedings of the IEEE 29th Real-Time and Embedded Technology and Applications Symposium (RTAS), San Antonio, TX, USA, 9–12 May 2023 ; pp. 40–53.
93. Santos, A.; Martins, J.; Sousa, J.; Rodríguez, M.; Pinto, S. Let's Get Physical: Rethinking the Static Partitioning Hypervisor Architecture for an MMU-less Memory Model. *Authorea* **2025**. [[CrossRef](#)]
94. Pinto, S.; Martins, J.; Rodriguez, M.; Cunha, L.; Schmalz, G.; Moslehner, U.; Dieffenbach, K.; Roecker, T. RISC-V Needs Secure 'Wheels': The MCU Initiator-Side Perspective. *arXiv* **2024**, arXiv:2410.09839.
95. IEC 61508-1:2010; Functional Safety of Electrical/Electronic/Programmable Electronic Safety-Related Systems—Part 1. International Electrotechnical Commission: Geneva, Switzerland, 2010.
96. Munir, A. Safety Assessment and Design of Dependable Cybercars: For today and the future. *IEEE Consum. Electron. Mag.* **2017**, *6*, 69–77. [[CrossRef](#)]

Disclaimer/Publisher's Note: The statements, opinions and data contained in all publications are solely those of the individual author(s) and contributor(s) and not of MDPI and/or the editor(s). MDPI and/or the editor(s) disclaim responsibility for any injury to people or property resulting from any ideas, methods, instructions or products referred to in the content.