Special Session: Emerging Architecture Design, Control, and Security Challenges in Software Defined Vehicles

Aya El-Fatyany[1,2], Xiaohang Wang[1], Parasara Sridhar Duggirala[3], Samarjit Chakraborty[3]
Sudeep Pasricha[4], Amit Kumar Singh[5]
Zhejiang University[1], Menoufia University[2], UNC Chapel Hill[3], Colorado State University[4], University of Essex[5]
{ayaelfatyany,xiaohangwang}@zju.edu.cn, psd@cs.unc.edu, samarjit@cs.unc.edu,
sudeep@colostate.edu, a.k.singh@essex.ac.uk

***Abstract*—Software Defined Vehicles (SDVs) represent a paradigm shift in the automotive industry, where vehicles are increasingly controlled and managed through software, while relying less on mechanical and hardware components. While this allows considerable flexibility in the introduction of new "smart" features and fast tracks innovations in multiple domains, it also creates new challenges and opportunities in architecture design, control, and security. By adopting modular architectures, adaptive control strategies, and robust security measures, SDVs can pave the way for a safer and more efficient future of transportation. In this paper, we cover perspectives from both, industry and academia, in this area. They provide embedded systems researchers an overview of recent developments and emerging challenges in SDV from the perspective of architecture design, control, and security. The emerging challenges also set the foundations for future research in this domain.**

***Index Terms*—Software defined vehicles (SDVs), Architecture design, Control, Robustness, Security.**

I. Introduction

In the automotive industry, Software Defined Vehicles (SDVs) represent a transformative shift, where the core functions and features of vehicles are increasingly controlled and updated through software rather than traditional hardware modifications or mechanical components. This software-based approach leverages advanced computing, connectivity, and over-the-air updates to enhance vehicle capabilities, including user experience, performance and safety. Even after vehicles leave the production line, such a software-based approach enables manufacturers to continuously improve and personalize vehicles, offering unprecedented flexibility and adaptability in response to changing technological advancements, customer needs and regulatory requirements. As vehicles evolve into more sophisticated and interconnected platforms, software-defined approaches are expected to play a pivotal role in the future of mobility.

While SDVs bring enhanced flexibility and customization, they also introduce an array of challenges [1] in architecture design, control, and security. As vehicles become more reliant on software to manage various aspects like engine performance, braking systems, infotainment and autonomous driving capabilities, the underlying architecture must be cautiously designed so that various subsystems can be seamlessly integrated [2]. This includes addressing issues like real-time data processing, interoperability between legacy systems and new software modules, and ensuring robust communication – perhaps using wireless networks in the future [3], [4] – within the vehicle. For example, automotive control systems that were once largely mechanical or analog, now they require sophisticated algorithms and real-time processing to efficiently manage the interactions between various vehicle components. This complexity demands for thorough testing [5], verification [6], and debugging [7] processes to ensure that the software functions as intended under all conditions, including safety-critical situations.

Similarly, robust perception in SDVs relying on embedded systems presents a range of challenges [8] that must be overcome to ensure the reliability [9] and safety of these vehicles. The challenges include achieving processing efficiency, real-time data processing and sensor integration and calibration. Moreover, the increasing connectivity of SDVs to external networks and cloud services introduces significant cybersecurity challenges [10]. This makes vehicles as potential targets for hacking, data breaches, and other malicious activities. Therefore, protecting the vast amounts of data generated by and transmitted within SDVs is of paramount importance. The security aspect is further complicated by integrating third-party applications and services, which is essentially to designing an advanced vehicle. This necessitates stringent control measures and secure software update mechanisms [11] to prevent unauthorized access and ensure the integrity of the software ecosystem. In this rapidly evolving landscape, addressing these emerging challenges in architecture design, control, and security is crucial to realizing the full potential of SDVs while safeguarding the safety, privacy, and trust of users. This paper discusses some of the aforementioned emergent challenges and solutions being debeloped to address them.

II. In-Vehicle Architectures in the Age of SDVs

Over the past years, automotive in-vehicle architectures are evolving from having many distributed ECUs (electronic control units) that are connected by multiple communication buses and gateways, to more centralized architectures where fewer powerful ECUs are assigned all computational tasks. This move has multiple advantages, but also poses several challenges. Instead of point-to-point connections between ECUs, there is now an emphasis on flexible *service-oriented architectures* [12], where different services hosted on different ECUs are advertised, and clients are dynamically matched

to services at startup time. This allows adding new services and clients much more flexibly. In previous architectures, adding new functions necessitated ensuring the availability of sufficient computation and communication resources and guaranteeing that the timing behavior of existing tasks are not disrupted [13]. While service-oriented architectures provide flexibility, they also add to additional delays that need to be mitigated – both through architectural innovations, new timing analysis techniques, new methods for application scheduling [14], [15] and resource allocation [16], and new ways of designing automotive control software [17].

The core idea behind SDVs is to decouple software architectures from the hardware platform on which the software applications run. Such separation has happened in many other domains like smartphones, where Android or iOS allows application developers to independently develop software without getting into the hardware intricacies of different smartphone architectures. Similarly, software-defined networks allow the flexible development of software-based networking services without requiring network addresses or firewalls to be reconfigured. The goal of SDVs is to introduce such flexibility in the automotive domain. This is to some extent already allowed by standards like AUTOSAR, but more innovations are needed to better decouple automotive hardware architectures from the software design decisions.

The evolution of automotive in-vehicle E/E (electrical and electronic) started with distributed architectures where each function was mapped to a different ECU and ECUs communicated using signal-oriented protocols. As the number of ECUs and the length of communication cables started growing to levels that were not sustainable, there was a push to consolidate. This led to the evolution of domain-oriented architectures where domain controllers consolidated multiple domain functions that were previously distributed. Currently, zone-oriented architectures are being considered, that distribute each control application even much further. Such architectures connect a diverse set of sensors and actuators that are in close physical proximity, to a zone ECU or zone controller. Cross-zone communication is enabled via a high-bandwidth backbone, e.g., using Ethernet. Also, few powerful vehicle computers or central ECUs are used to implement most of the main computations and decision making, which can be OEM-specific. Zone controllers connect to the vehicle computing cluster also via high-bandwidth automotive Ethernet. Such an architecture enables significant reduction in the length and weight of a car's wiring harness, which is currently one of the heaviest car components, along with the engine and chassis, and also one of the most expensive ones. This reduction is because sensors and actuators now need not be connected to different individual ECUs or even domains, and can instead be connected to the nearest zone ECU. However, such architectures also imply that signals belonging to the same control application will now have to traverse across multiple communication zones. The corresponding signal delays, which depend on how communication buses and gateways are configured, will affect control performance. Further, functions from different control applications, that might even come from different suppliers, now share the same ECU. Such sharing is enabled by common interfaces defined by AUTOSAR. But the task mapping and scheduling decisions also open a bigger design space and play an important role in determining control performance. Robustness and composability are important requirements for such architectures – they enable new software components to be seamlessly added to the system without disrupting the timing behavior of existing applications. How to enable this using a mix of time- and event-triggered architectures [18] are open research problems that are currently being studied.

III. Automotive Control in the Era of SDVs

The algorithmic core of the millions of lines of software code in SDVs is made up of feedback control loops. Traditional workflows involve first designing the control strategy, followed by generating software code corresponding to the designed strategy, and finally implementing the code on a distributed embedded platform as found in modern automotive E/E architectures [19]. This workflow follows the principles of "separation of concerns" and has worked well for *federated automotive architectures* with independent or loosely-coupled ECUs (electronic control units) that supported the "one function per ECU" paradigm. But with the SDV paradigm and integrated architectures, where sensing, control, and actuation tasks are spread across multiple ECUs and share such ECUs with other software tasks, the traditional workflow of *first designing and then implementing* approach to control software development is breaking down. This is because designing control strategies without accounting for implementation platform details, freezes the design and does not allow exploiting the many different ways in which control software tasks may be partitioned, mapped and scheduled on modern in-vehicle E/E architectures. Each of these options results in different timing behaviors experienced by the control software, and accuracy of machine learning based perception, which necessitates adjusting the control strategy and its parameters – something no longer possible if the control design is already frozen. This also results in a large semantic gap between models of controllers and their implementations, which necessitates increased testing and debugging. In this paper we discuss multiple emerging techniques for addressing these challenges, which are prompting new paradigms for automotive control design and implementation, including the use of machine learning based approaches.

A. Framework & Tools for Control/Architecture Co-Design

In order to account for the different timing characteristics associated with the different implementation choices of a control application, we have designed a *control/architecture co-design* framework [20], [21], [22] and a corresponding toolchain [23] to implement it. The input to such a framework is a controller *template* that specifies the high-level structure of the controller, without all the parameters such as the sampling period and the controller gain values being specified. The second input to this framework is a set of different implementation constraints, such as the dependencies between the tasks of each controller sharing implementation resources,

the maximum utilization of each resource (processors and buses), and scheduler templates. The framework then explores different implementation options for the control tasks [24], [25], such as task mappings, and parameters of the scheduler templates on the processors and the buses (e.g., task priorities, deadlines, or slot lengths for time-triggered schedules). Each of these implementation options represent different timing behaviors experienced by each controller. In order to maximize control performance, the controller parameters need to be optimized for the specific timing behavior in question. Our co-design framework automatically synthesizes such controller parameters for each implementation choice being explored. In this process, the *best combination* of controller parameters and controller implementation is identified. The metrics that may be optimized in this process can be resource utilization, maximizing the number of controllers that can be implemented on a given platform, or control performance metrics such as stability, settling time, or peak overshoot.

To implement this framework, we have used a set of standard tools and connected them to form a toolchain. Depending on the specific setup at hand, the toolchain can include **(i)** a set of Simulink template blocksets to model control applications, and a set of tool-specific blocksets for architecture configuration (e.g., tools from Mentor Graphics support blocksets specific to their tools); **(ii)** code to automate the flow between controller design and architecture configuration phases – these include specification extraction, re-configuration of the control models with the calculated values of the control parameters, and synthesizing implementation architecture parameters, e.g., ECU and bus schedules; **(iii)** code for *multi-objective design space exploration* and co-optimization to simultaneously synthesize control and architecture parameters while accounting for different design tradeoffs between e.g., quality of control and resource utilization.

More concretely, we start with (1) *specification modeling*, where the Simulink *blocksets* are used to partially specify control algorithms according to design specifications, e.g., plant descriptions and control objectives. In addition, the implementation architecture is also modelled, using a collection of tools with each responsible for a part of the architecture. For example, a FlexRay network and a CAN network may be specified using different tools. This is followed by (2) *specification extraction*, where a *Parse* tool extracts the necessary information from the partially specified controllers and architecture description files. For this, we have used the AUTOSAR FIBEX (Field Bus Exchange) XML format that is supported by all automotive design tools. The extracted information was organized into a collection of different files. Next, in a (3) *design space exploration* phase, a *co-optimization* tool jointly synthesizes all controller and architecture parameters. This is where the bulk of our research has feed into, and used different optimization tools like CPLEX and Gurobi[1]. The output of this phase is a visual representation of feasible design configurations showing different tradeoffs between multiple design objectives, i.e., a *pareto front*. Next, in (4) *parameter write-back*, the designer selects a specific parameter set from the different tradeoffs, and the *configure* tool writes these parameters back to partially-specified control and architecture models in order to complete their configuration and ensure the necessary compatibility between them. The fully specified system models can now be used to (5) generate code or binary files using the Simulink code generator and they can also be used to generate, e.g., bus controllers using tools like the Mentor Capital Systems Network. These are then flashed onto ECUs, thereby completing the entire design process. These ECUs are next used in HiL (hardware-in-the-loop) simulations to verify that the system works as intended [26], [27].

[1] ibm.com/analytics/cplex-optimizer, gurobi.com

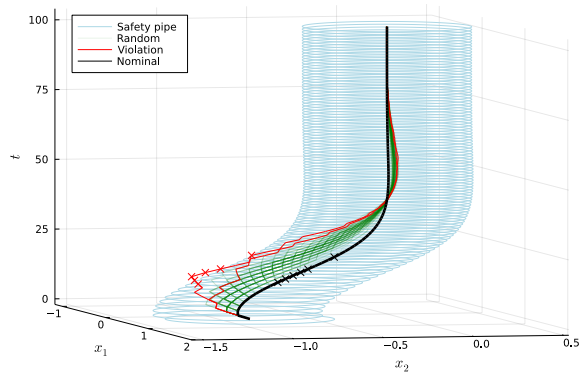

Fig. 1. Safe & unsafe behaviors due to deadline misses [28].

B. System-level Safety Driven Controller Implementations

Resource sharing and distributed implementations — the two main characteristics of SDVs — result in large delays that can adversely impact control performance. Traditional design workflows rely on "design followed by implementation," where control engineers design controllers and determine their parameters like sampling periods and tolerable sensor-to-actuator delays. This is followed by embedded systems engineers implementing the controllers in software while ensuring that design-time parameters like sampling periods and delays are respected. In this process, *deadlines* have become synonymous with *safety*, and a control task missing its deadline is considered a safety violation. While this approach allows a beneficial separation between design and implementation, it is inflexible and overly conservative.

To address this, we have viewed the satisfaction of deadlines to be a "secondary property," and instead attempted to focus on the satisfaction of *system-level safety properties.* Such system-level safety properties better reflect actual safety concerns, and they might continue to be satisfied even when the deadlines of software tasks are occasionally violated. This affords more flexibility during implementation with respect to meeting timing constraints. In particular, this allows better distribution of sensing, control, and actuation tasks on an architecture, and also allows more control tasks to be implemented or "packed" on the same architecture. While there could be different notions of what constitutes system-level safety, we have recently studied one where we assume a safety pipe around the ideal trajectory of the system in its state space and refer to any trajectory as safe as long as it stays within the safety pipe. This is shown in Figure 1 based on the model of an automotive electric steering system [28]. Here, the solid black line shows

the ideal system trajectory, viz., the evolution of the closed-loop system in the (x_1, x_2) state space, when the control task always meets its deadline (which may not be feasible in a real implementation, as in SDVs). The envelope around the black line is the safety margin, representing system behaviors that are considered to be safe. The system trajectories shown in the red and green lines are obtained when the control task misses some of its deadlines during the evolution of the system, as such deadline misses cause the dynamics of the system to alter. Because of such deadline misses, control inputs necessary to actuate the system are not available and depending on the policy in place, an old control input may be applied. As expected from the definition of the safety pipe, the red trajectories violate the safety property and the time instants where the violations occur are marked with "×". The corresponding times have been marked with a black × on the ideal trajectory.

To be able to exploit this notion of system-level safety and allow more flexible timing behaviours, note that the dynamics of the closed-loop system, i.e., the system trajectories obtained in Figure 1, depend on (i) the task deadline hit/miss patterns, (ii) which control input is applied when a control task misses its deadline and therefore the intended control input is not available, and (iii) how is the incomplete control task handled, e.g., whether it is killed or is allowed to execute beyond its deadline, which impacts the load on the system and therefore future task deadline misses. In our recent work [28], we have studied different possibilities with regards to (ii) and (iii), that were proposed in [29]. There exists many standard techniques to check whether the closed-loop system becomes unstable because of deadline misses [30]. Hence, our work has focused on systems that remain stable, but we intend to check whether their dynamics deviate too much from the ideal behavior to be deemed unsafe, viz., whether they go outside the safety pipe. Second, we are interested in identifying which timing behaviors or deadline hit/miss patterns are safe. Finally, how can we exploit this notion of system-level safety to implement more control tasks [31] on an implementation platform that would otherwise not be possible?

For a specific deadline hit/miss pattern and a specified strategy to handle deadline misses, it is straightforward to check whether the safety property is satisfied over any finite time horizon. This is done by "simulating" a run of the closed loop system for this deadline hit/miss pattern. But if the number of timing behaviors or deadline hit/miss patterns allowed by an implementation platform is very large — which is the case in reality — then it is no longer feasible to check the dynamics of the system for each of these patterns [32]. To get around this, we developed a safe but approximate reachability analysis technique [28], [33]. Here, *safe* implies that if our reachability analysis technique certifies a set of deadline hit/miss patterns to satisfy the specified safety property, then all system trajectories resulting from these deadline hit/miss patterns are guaranteed to remain within the safety pipe. The technique is approximate because if it deems a set of deadline hit/miss patterns to be unsafe then the system trajectories resulting from them may or *may not* lie outside the safety pipe, i.e., the safety property may not be violated.

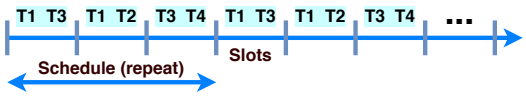

Fig. 2. Scheduling control tasks with deadline misses.

Using the above-mentioned approximate reachability analysis, we now briefly outline a technique for synthesizing schedules to pack more control tasks on an implementation platform compared to what would be possible if all task deadlines were to be met. For this, consider a set of four control tasks $T_1, \ldots, T_4$ that need to be scheduled on a single processor. This technique can be extended to a distributed implementation platform with multiple processors and buses. Time is partitioned into equal-sized slots (see Figure 2), and for simplicity we assume that this slot size is also equal to the sampling period of each of the control tasks. For the purpose of illustration, we assume that the Worst-Case Execution Time (WCET) [34], [35] of each of the control tasks $T_1, \ldots, T_4$ is such that at most two of them may be scheduled in each time slot, and if a task T_i is not scheduled in a particular slot then it misses its deadline. As outlined above, when a control task misses its deadline, based on the policy in place, the previous control input may be applied, causing the dynamics of the system to deviate.

The schedule for the tasks $T_1, \ldots, T_4$, shown in Figure 2 satisfies the scheduling constraint that no more than two tasks can be scheduled in one time slot. It may be observed that the schedule for any task T_i may be represented as a binary string. In Figure 2, the one for T_1 is $110110\ldots$, that of T_2 is $010010\ldots$, T_3 is $101101\ldots$, and finally, that of T_4 is $001001\ldots$. Here, 1 denotes the deadline in a particular time slot being met and 0 denotes a deadline miss. Given a concrete schedule of this form, we can check whether it satisfies the scheduling/resource constraint and the safety constraint. The safety constraint is validated by checking whether a run of the system with the deadline hit/miss pattern dictated by the schedule results in a trajectory that is contained within the safety pipe. To check the safety of a *set* of binary strings (deadline hit/miss patterns) and not one concrete string, we use the approximate reachability analysis outlined earlier. When such sets of binary strings constitute regular languages, and satisfy a specified system-level safety property, we have shown in [31] that using automata-theoretic techniques we can synthesize schedules of the form in Figure 2. In particular, given a regular language for each task T_i to be scheduled, where the language represents a class of safe timing behaviors, it is possible to synthesize a schedule for all of these tasks that satisfy the resource constrains on the system, e.g., that no two tasks may be scheduled in each slot. Extensions to more general setups and resource constraints have been recently studied in [36].

Hence, as delays become larger with in-vehicle architectures supporting SDVs, and there is more pressure to accommodate the growing volume of software in vehicles, re-thinking how control algorithms are designed and implemented is necessary. We have discussed the possibility of introducing a new notion of safety, viz., one that considers system-level properties

instead of deadline misses that may or may not impact system-level behavior. This approach may be extended in many different ways, including to what kind of guarantees should security [37], [38] or machine learning (ML) components provide for safety certification [39], [40], [41].

IV. Emerging Robustness: Robust Perception with Embedded Systems in SDVs

In emerging SDVs, robust environmental perception with embedded systems is critical to informing control and other operational components that directly impact driving behavior [42]. Robustness in the context of SDVs refers to multi-faceted objectives that include real-time, safety-critical, secure, and accurate perception-driven operation, despite the presence of multiple uncertainties, noise sources, and threat vectors.

One of the first challenges that must be overcome for robust perception in SDVs relates to supporting real-time communication among various distributed sensors and computing units within the vehicle. Any violation of real-time deadlines associated with data transmission and processing can prevent timely perception. In emerging SDVs, time-triggered network protocols such as automotive Ethernet and Flexray are used to provide stronger real-time guarantees than traditional event-driven protocols such as CAN. However, time-triggered automotive transmissions are still susceptible to jitter, which is the stochastic delay-induced deviation from the actual periodicity of a message. At a high level, jitter can be classified into either bounded (deterministic) jitter or unbounded (random) jitter [43]. Bounded jitter is a periodic variation due to the systematic occurrences of certain events in the system, such as queuing of messages, clock jitter, etc., whose peak-to-peak value is bounded. Unbounded jitter is an unpredictable timing noise whose peak-to-peak value is not bounded, e.g., due to thermal noise in an electrical circuit that can delay task executions or message transmissions, external disturbances, etc. Such random jitter is very hard to predict, and can severely affect system performance, with catastrophic consequences in some cases, e.g., when the airbag deployment signal from the impact sensor to the inflation module gets delayed due to jitter. In [44], a novel framework called JAMS-SG was proposed to overcome the impact of random jitter in time-triggered vehicle networks. JAMS-SG integrated jitter in the early design phase, by performing jitter-aware frame packing (packing of different signals from an ECU into messages) and used a novel SA+GRASP heuristic for the synthesis of jitter-aware design time communication schedules. At the same time, to cope with unexpected jitter variations at runtime, the framework integrated Multi-Level Feedback Queues (MLFQs) with an intelligent runtime scheduler that opportunistically packs jitter affected time-triggered and high priority event-triggered messages for jitter-resilient communication. A custom frame format was also introduced to solve the addressing and segmentation problem associated with packing multiple jitter-affected messages. In experimental analysis across varying jitter conditions, JAMS-SG was able to achieve lower response times compared to the state-of-the-art approaches, and more importantly, without missing any message deadlines. The framework can be adapted to a variety of time-triggered protocols with minimal changes. In [45], this framework was extended to integrate lightweight symmetric key cryptography to further prevent data corruption (either malicious or noise related) over the vehicle network, while ensuring that real-time constraints were not violated.

The actual design of the robust perception system for SDVs entails solving multiple complex problems. First, the optimal set of sensors must be selected from a heterogeneous suite of sensors, e.g., cameras, radars, lidars. After the number and type of each sensor has been selected, for each of these sensors, their location and orientation on the vehicle must be determined, to maximize environmental coverage in the vehicle field of view. The combination of these sensors must enable high object detection rates and low false positive detection rates, for which deep machine learning techniques have been found to be particularly effective [46]. Additionally, such techniques must meet the stringent latency requirements, onboard memory capacity, and computational complexity in vehicle contexts, which is challenging. Lastly, as the position of obstacles and traffic are highly dynamic in real-world scenarios, so after detection of an object, tracking is required to predict its new position. Due to noise from various sources there is an inherent uncertainty associated with the measured position and velocity of an object. This uncertainty can be minimized by using sensor fusion algorithms. An important challenge with sensor fusion algorithms is that the complexity of tracking objects increases as the objects get closer, due to a much lower margin for error (uncertainty) in the vicinity of the vehicle [47].

The VESPA framework [48] represents one of the first frameworks for automated sensor placement and orientation of sensors in SDVs. The work explored various design space exploration algorithms including Simulated Annealing and Greedy Random Adaptive Search Procedure (SA + GRASP), Genetic Algorithm (GA), and Particle Swarm Optimization (PSO) to determine the best algorithm for the problem. Given a specific vehicle and its dimensions, a candidate set of heterogeneous sensors, and ADAS features to be supported (e.g., to meet a specific SAE autonomy level). It was shown that the GA algorithm provided the best results, generating sensor selections, placements, and orientations to meet perception targets. The VESPA framework was shown to optimize perception performance across multiple ADAS features for the 2019 Chevrolet Blazer and 2016 Chevrolet Camaro vehicles. The PASTA framework extended this work to not only determine sensor placement and orientation, but also co-optimize them with a deep learning based object detector design, and sensor fusion algorithm [49]. The heterogeneous sensor library, object detection model library (including YOLOv3, YOLOv4, SSD, R-CNN, Fast R-CNN, and Faster R-CNN), sensor fusion algorithm library (including Kalman filter, Extended Kalman filter, and Unscented Kalman filter), and physical dimensions of the vehicle model are provided as inputs to the framework. An algorithmic design space exploration is used to generate a perception architecture solution which is evaluated based on a cumulative score from performance metrics relevant to the autonomy level being targeted. Five design space search

exploration algorithms were explored as part of the framework: genetic algorithm (GA), differential evolution (DE), firefly algorithm (FA), particle swarm optimization (PSO) and the hybrid genetic algorithm–particle swarm optimization (GA-PSO). Experimental analysis with a BMW Minicooper and Audi-TT vehicles demonstrated that the GA algorithm based approach provided the best results. The perception architecture performance with PASTA outperformed the VESPA framework, due to PASTA's more comprehensive co-optimization across a larger number of key components of the perception architecture.

A fundamental challenge with deploying perception architectures, such as the ones generated by the VESPA and PASTA frameworks, in SDVs, involves ensuring that the associated algorithms can execute in real-time on resource constrained embedded platforms. One of the most resource intensive components in any perception architecture is the vision-based object detector that is utilized in most emerging SDVs. This is a compute and memory-intensive task involving both classification and regression, and typically performed with machine learning models. One type of machine learning based object detector is a two-stage detector that employs a region proposal stage and subsequent object classification stage. The region proposal stage often consists of a Region Proposal Network (RPN) which proposes several Regions of Interest (ROIs) in an input image. These ROIs are used to classify objects in them. The objects are then surrounded by bounding boxes to localize them. Examples of two-stage detectors include R-CNN, Fast R-CNN, and Faster R-CNN. In contrast to two-stage detectors, single-stage detectors use a single feed-forward network which involves both classification and regression to create the bounding boxes to localize objects. Some examples of single-stage detectors are YOLOv5 (You Only Look Once), RetinaNet, YOLOR, and YOLOX. While single-stage detectors are lightweight and faster than two-stage detectors, they are less accurate. Unfortunately, even the lightweight single-stage detectors are compute and memory intensive, and it remains challenging to efficiently deploy them on embedded and IoT platforms. The R-TOSS framework [50] provides a possible solution to this challenge. This framework performs automated model pruning on object detectors to reduce their latency, memory footprint, and energy overheads. Unlike traditional pruning algorithms that can generally be classified as unstructured pruning or structured pruning, the framework explores an unconventional semi-structured pruning approach, involving applying specific geometric kernel patterns to prune convolutional kernels and associated connectivity in object detector models. The framework also proposed and approach for reducing computational cost of iterative pruning by using depth first search to generate parent-child kernel computational graphs, to be pruned together. Additionally, a novel pruning technique to prune 1×1 kernel weights was proposed to increase achievable model sparsity. A detailed comparison against multiple state-of-the-art pruning approaches showed the effectiveness of the novel R-TOSS framework, outperforming several state-of-art pruning frameworks. Interestingly, the framework was able to increase the mAP of the object detectors compared to the mAP of the baseline models, and even more impressively, achieve these results without any compiler optimization or additional hardware requirements. Experimental results on the JetsonTX2 embedded board showed that the R-TOSS framework resulted in a model compression rate of 4.4× on the YOLOv5s and 2.89× on the RetinaNet object detectors, while outperforming the original model as well as several state-of-the-art pruning frameworks in terms of accuracy and inference time.

As the perception system in modern SDVs utilizes a large number of sensors and also relies on extensive external communication with roadside infrastructure and other vehicles, it possesses a large attack surface that can be exploited by attackers [51]. It is possible to compromise one or more sensors or communication interfaces remotely or via physical tampering to launch data integrity attacks that can corrupt data and data confidentiality attacks that can steal sensitive information. Thus, the deployment of perception architectures on embedded platforms in SDVs must include smart intrusion detection systems (IDS). An IDS is extremely crucial and often the last line of defense when an attacker breaks through defense mechanisms. Traditional IDS solutions relied on firewalls or rule-based (non-AI-based) systems to detect cyber-attacks. However, they are not effective against detecting sophisticated automotive attacks [51]. Recently, machine learning based IDS have emerged for SDVs. The INDRA IDS proposed in [52] is an example of such an approach, making use of a gated recurrent unit (GRU)-based recurrent autoencoder neural network for intrusion detection in SDVs. This IDS learns the normal operating behavior of the system by reconstructing the input attack-free data during training at design-time. During inference at runtime, the IDS observes the deviation from the learned normal system behavior to detect attacks in SDVs. This approach showed promising results when it was deployed on automotive embedded hardware platforms and tested under several attack scenarios. In [53], an unsupervised LSTM-based encoder-decoder neural network was proposed that integrates a novel self-attention mechanism to learn the characteristics of normal data. The attention mechanism enhanced the ability of the model to focus on the important hidden state information from the past. A one-class support vector machine (OCSVM)-based classifier was trained and deployed together with the self-attention model at runtime to detect attacks. This approach demonstrated superior performance compared to various state-of-the-art statistical, proximity-based, and machine learning based IDS approaches under different attacks scenarios, across multiple metrics: accuracy, F1 score, false-positive rate (FPR), and receiver operating characteristics (ROC) curve-area under the curve (AUC). A temporal convolutional neural attention (TCNA) network-based lightweight IDS called TENET was described in [54], to learn very long-term dependencies between messages transmitted over an in-vehicle network. This was used to effectively learn the normal system behavior of the vehicle at design time. A decision tree (DT)-based classifier was used to learn the model deviations that correspond to the normal vehicle operation at design time. At runtime, the trained TCNA and DT models were used to detect anomalous deviations using the deviation score metric, to detect attacks. This technique outperformed all the comparison works, including INDRA, in the Matthews correlation coefficient (MCC)

metric. Such IDS frameworks will become increasingly crucial to realize robust perception architectures in future SDVs.

V. Emerging Security: Covert Channel and In-vehicle Network Spoofing Attacks on Embedded Systems in SDVs

Covert channels and in-vehicle network spoofing attacks pose significant risks to the safety, privacy, financial stability, regulatory compliance, and consumer trust associated with SDVs. In covert channel attacks, the malicious nodes/programs use undetected channels, e.g., heat transfer, vibration, inter-packet delay, light color, etc. to secretly transmit sensitive data in an unprotected manner. This type of attacks, together with vulnerability detection and side channel attacks, can leak sensitive data to violate privacy. We first discuss how typical covert channel attacks work at chip level and embedded systems in SDVs. The attack effects are analyzed and modeled. Detection methods will also be discussed to find these secret data transmissions along with the countermeasures to stop the information leakage. Finally, we will present our recent works on hacking the in-vehicle networks in a real car to control various components of the car, where attacks are based on reverse engineering the CAN protocol followed by injecting car-control commands.

A. Covert Channel Attacks on Embedded Systems in SDVs

Software-Defined Vehicles (SDVs) leverage high-performance many-core System on a Chip (SoC) resources to facilitate autonomous driving and various intelligent applications. These SoCs integrate specialized units for AI and robotic algorithms, multi-core CPUs for general tasks, and many-core CPUs for real-time processing [55]. A heterogeneous architecture is essential for balancing the processing power and energy efficiency in advanced embedded systems for autonomous vehicles. Nonetheless, many-core SoCs face challenges, such as thermal covert channels.

Thermal covert channel (TCC) attacks present a significant security challenge for many-core chips. These attacks are closely related to the transmission rate and bit error rate (BER) of the TCC, which are influenced by the transmission characteristics of the thermal signals and the modulation, encoding, and multiplexing techniques used. The versatility of TCCs was demonstrated in a study by [56], which employed various modulation and line-coding methods, as well as multiplexing techniques. The authors compared the performance of 13 different TCCs in terms of BER and transmission rate, which allowed them to quantitatively assess the potential damage and develop suitable countermeasures to address TCC attacks. Understanding the maximum transmission throughput of different TCC designs is crucial for identifying potential vulnerabilities and implementing effective mitigation strategies. The research in [57] concentrated on three primary areas that hinder the efficiency of thermal covert channels: signal generation at the source end, signal decoding at the destination end, and end-to-end communication protocols. This study introduced several techniques to improve the capacity of thermal covert channels by overcoming thermal interference. Specifically, in this study, data in a thermal covert channel were encoded and represented using a novel thermal signaling scheme, where the logic value, 0 or 1, modulates the duty cycle of the thermal signals. Additionally, the study demonstrated that selecting an appropriate transmission frequency can significantly reduce thermal interference. Furthermore, it proposes a robust end-to-end communication protocol for reliable communication. The proposed attack can reduce the Bit Error Rate (BER) by 75% and enhance the transmission rate by 370%. Therefore, the proposed attack can be utilized to transmit passwords secretly on a chip with a stable transmission rate of 160bps and a BER as low as 0%. The study described in [58] aimed to investigate the potential security threats in multi-core systems by measuring the temperature. The authors sought to regulate fan speed and reduce the temperature of the system to an appropriate level as a protective measure against these attacks. The results indicate that a substantial decrease in temperature is correlated with a decrease in power consumption.

The necessity for a reliable and confidential method for transmitting sensitive data over an untrusted network-on-chip (NoC) integrated into a system-on-chip (SoC) design has arisen because of the potential risk of compromised data integrity and security. In [59], a novel approach was presented that ensures secure data transmission over an untrusted NoC. This scheme encodes binary data as inter-packet delays between packets traveling between a source-destination pair. The analytical model developed in this study was utilized to determine the maximum data transmission rate of this IPD-based communication channel. To enhance the undetectability and robustness of the proposed data transmission scheme, a new block coding method and communication protocol were proposed. The IPD channel offers several advantages over competing TCC, cache covert channel, and circuit-based encryption schemes, including a lower packet error rate (PER) and overhead and higher throughput, making it a suitable option for secure data transmission in unsecured NoC-based systems.

Owing to increasing security concerns posed by thermal covert channel (TCC) attacks on many-core systems, multiple detection methods based on threshold levels have been developed. However, these methods are insufficient for detecting TCCs using advanced signaling and specific modulation techniques, as highlighted in [60]. To address this issue, this study proposes a detection method based on pattern classification that can effectively counter improved stealthy TCCs that use reduced signal amplitude to evade threshold-based detection. This proposed method achieved a detection accuracy of 99% for both the baseline TCC and the improved stealthy TCC. After employing the DVFS-based countermeasure on the detected CPU cores, the PERs of both types of TCCs were higher than 70%, but with a low runtime and low energy. The proposed detection method and DVFS-based countermeasure are low-complexity and low-overhead, making them well-suited to work together to thwart any known TCC attacks with ease.

A previous study [61] demonstrated that thermal signals combined with noise using direct sequence spread spectrum (DSSS) modulation exhibited no detectable amplitude differ-

ence for the detector. This makes DSSS-based TCC highly challenging to detect and requires innovative approaches owing to the transmission of noise-like signals. This study proposes isolating useful signal components by eliminating low-frequency noise and irrelevant signals, followed by a threshold-based detection method to identify potential TCC. Upon detecting a DSSS-based TCC attack, a countermeasure involving a strong noise injection to jam the TCC can be employed. The authors reported that the proposed detection scheme achieved a high accuracy of up to 89% for detecting TCC without DSSS- and DSSS-based TCC.

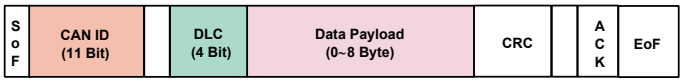

Fig. 3. CAN Bus Frame.

B. In-vehicle Network Attacks and Countermeasure

Contemporary automobiles are equipped with various electronic systems that provide new and convenient functions for drivers and passengers. To implement advanced driver assistance systems (ADAS), such as lane keeping assist (LKAS), anti-lock braking systems (ABS), and smart cruise control (SCC), vehicles rely on electronic control units (ECUs) instead of manual control. Consequently, many ECUs have been installed in contemporary vehicles [62]. Controller area networks (CAN) have emerged as the de facto standard for communication among ECUs in vehicles. As shown in Fig.3, a standard CAN frame comprises several fields including the Start of Frame (SOF), Arbitration Field (CAN ID), Control Field, Data Field (payload), CRC field, knowledge field (ACK), and End of Frame (EOF). CAN networks can be connected to external systems through connected and autonomous vehicle technologies. However, CAN lacks security measures such as confidentiality, authentication, and access control, enabling attackers to access the CAN bus and inject packets that manipulate critical functions, such as emergency braking, acceleration, and steering control.

An attacker can potentially exploit external interfaces, such as an OBD-II diagnostics port, telematics unit, or in-vehicle infotainment (IVI), to launch an attack on a real car [63]. Hackers can carry out various types of critical attacks, including gaining knowledge about the architecture and behavior of connected electronic control units (ECUs) in the vehicle. However, these attacks do not require specific prior knowledge. Each manufacturer employs its own set of identifiers for the same functions, which are encoded in the CAN matrix, and may not be constant for every model [64]. Additionally, other types of attacks, such as Denial of Service (DoS), Fuzzing, Spoofing, Malfunction, can occur.

Furthermore, CAN networks face significant challenges in protecting against remote attacks, owing to their long-range connectivity. Covert channels, which are hidden communication paths, have emerged as potential threats and protective measures to these networks. Specifically, the use of covert channels in Controller Area Networks (CAN) has been explored for their ability to enhance vehicular message authentication procedures while maintaining the system performance.

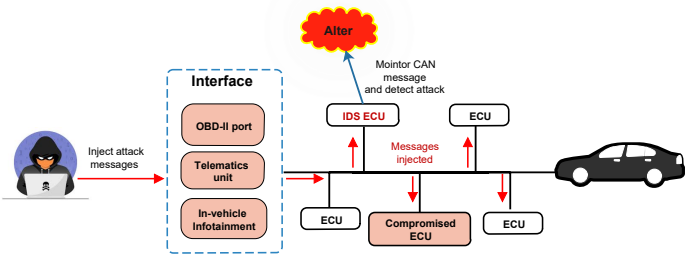

Fig. 4. CAN Bus IDS Architecture.

In a study conducted by researchers [65], the potential for benign defensive uses of covert channels to support vehicular message authentication mechanisms was investigated as a transparent resource-conserving approach to automotive network security. The authors provide the first comprehensive evaluation of covert channels in Controller Area Networks (CAN) with respect to the achievable bandwidth and reliability of covert communication. Researchers have identified timing-based covert channels as potential candidates for developing a supplementary nonce synchronization channel that can enhance the resilience of existing authentication mechanisms to message loss.

[66] proposed TACAN, a secure method for authenticating ECUs on a legacy CAN bus using covert channels without modifying the CAN protocol or increasing traffic. TACAN utilizes a centralized Monitor Node and employs IAT-based, LSB-based, and hybrid covert channels for ECU authentication. The study reported that the TACAN-based detector accurately identified CAN bus attacks with a high detection rate and a false alarm rate below 0.33 percent when the detection threshold was set to 2 or higher.

Moreover, to prevent, detect, and mitigate CAN bus network attacks, an intrusion detection system (IDS) can be implemented between the external connection and CAN bus, as depicted in Fig.4. The IDS will filter out any malicious traffic that an attacker attempts to send into the CAN bus system, and it will generate alert messages.

In [67], a deep convolutional neural network (DCNN)-based intrusion detection system (IDS) was proposed to safeguard the CAN bus from cyber-attacks. The authors introduced a frame builder, which is a straightforward data-assembly module that directly uses bit-stream data from the CAN bus in the CNN. This module converts the CAN bus data into a grid-like structure that is compatible with the DCNN. Using the grid data frame instead of a feature vector, the CNN-based classifier can effectively learn temporal sequential patterns in CAN traffic data without additional preprocessing. The IDS was designed using a modified Inception-ResNet model originally developed for large-scale image classification by reducing the size and number of layers.

The study in [68] introduced HistCAN, an advanced intrusion detection system (IDS) for anomalies on CAN buses. HistCAN employs a hybrid convolutional neural network (CNN)-multilayer perceptron (MLP) structure to create a spatial-temporal representation and capture multidimensional data from CAN sequences. It also includes a historical information fusion module to capture long-term dependencies across the CAN ID series, enhancing detection accuracy and performance. Experiments revealed that HistCAN often

outperforms benchmark methods, achieving an impressive F1 score of 0.9954 in a fully self-supervised manner while meeting real-time requirements.

VI. Concluding Remarks

The paper discussed existing efforts and design challenges towards adoption of SDVs that represent a paradigm shift in the automotive industry. Various aspects of emerging architecture, control and security challenges were considered from both industrial and academic points of view, while reviewing state-of-the-art approaches. While this paper discussed architectural issues, control, robust perception, and security separately, they will increasingly need to be co-designed in order to better exploit the rich tradeoffs that exist between them. Further, as autonomous features continue to increase in modern cars there will also be interesting tradeoffs between resource requirements, design complexity, extensibility and certifiability that need to be studied in the future.

VII. Acknowledgments

We acknowledge financial support from the following: National Science Foundation grant CNS-2132385 and CPS-2038960, the National Natural Science Foundation of China under Grants 92373205 and 62374146, the National Key Research and Development Program of China No. 2023YFB4404404, CCF-Phytium Fund, the Key Technologies R&D Program of Jiangsu (Prospective and Key Technologies for Industry) under Grant BE2023005-2.

References

[1] U. D. Bordoloi *et al.*, "Autonomy-driven emerging directions in software-defined vehicles," in *Design, Automation & Test in Europe Conference & Exhibition (DATE)*, 2023.

[2] F. Sagstetter *et al.*, "Schedule integration framework for time-triggered automotive architectures," in *51st Annual Design Automation Conference (DAC)*, 2014.

[3] P. H. Kindt *et al.*, "Energy modeling for the bluetooth low energy protocol," *ACM Trans. Embed. Comput. Syst.*, vol. 19, no. 2, pp. 13:1–13:32, 2020.

[4] ——, "Neighbor discovery latency in ble-like protocols," *IEEE Trans. Mob. Comput.*, vol. 17, no. 3, pp. 617–631, 2018.

[5] A. Karimi and P. S. Duggirala, "Automatic generation of test-cases of increasing complexity for autonomous vehicles at intersections," in *13th ACM/IEEE International Conference on Cyber-Physical Systems, ICCPS 2022, Milano, Italy, May 4-6, 2022*. IEEE, 2022, pp. 1–11.

[6] P. Kumar *et al.*, "A hybrid approach to cyber-physical systems verification," in *49th Annual Design Automation Conference (DAC)*, 2012.

[7] D. Roy *et al.*, "Timing debugging for cyber-physical systems," in *Design, Automation & Test in Europe Conference & Exhibition (DATE)*, 2021.

[8] C. Hobbs *et al.*, "Perception computing-aware controller synthesis for autonomous systems," in *Design, Automation & Test in Europe Conference & Exhibition, (DATE)*, 2021.

[9] G. Georgakos *et al.*, "Reliability challenges for electric vehicles: from devices to architecture and systems software," in *50th Annual Design Automation Conference (DAC)*, 2013.

[10] P. Waszecki *et al.*, "Automotive electrical and electronic architecture security via distributed in-vehicle traffic monitoring," *IEEE Trans. Comput. Aided Des. Integr. Circuits Syst.*, vol. 36, no. 11, pp. 1790–1803, 2017.

[11] P. Mundhenk *et al.*, "Lightweight authentication for secure automotive networks," in *Design, Automation & Test in Europe Conference & Exhibition (DATE)*, 2015.

[12] E. Fraccaroli *et al.*, "Timing predictability for SOME/IP-based service-oriented automotive in-vehicle networks," in *Design, Automation & Test in Europe Conference (DATE)*, 2023.

[13] A. Masrur *et al.*, "Vm-based real-time services for automotive control applications," in *16th IEEE International Conference on Embedded and Real-Time Computing Systems and Applications (RTCSA)*, 2010.

[14] D. Roy *et al.*, "Goodspread: Criticality-aware static scheduling of CPS with multi-qos resources," in *41st IEEE Real-Time Systems Symposium (RTSS)*, 2020.

[15] S. Chakraborty and L. Thiele, "A new task model for streaming applications and its schedulability analysis," in *Design, Automation and Test in Europe Conference and Exposition (DATE)*, 2005.

[16] D. Roy *et al.*, "Tighter dimensioning of heterogeneous multi-resource autonomous CPS with control performance guarantees," in *56th Annual Design Automation Conference (DAC)*, 2019.

[17] W. Chang *et al.*, "OS-aware automotive controller design using non-uniform sampling," *ACM Trans. Cyber Phys. Syst.*, vol. 2, no. 4, pp. 26:1–26:22, 2018.

[18] M. Lukasiewycz *et al.*, "Modular scheduling of distributed heterogeneous time-triggered automotive systems," in *17th Asia and South Pacific Design Automation Conference (ASP-DAC)*, 2012.

[19] D. Roy *et al.*, "Semantics-preserving cosynthesis of cyber-physical systems," *Proc. IEEE*, vol. 106, no. 1, pp. 171–200, 2018.

[20] R. Schneider *et al.*, "Constraint-driven synthesis and tool-support for Flexray-based automotive control systems," in *9th International Conference on Hardware/Software Codesign and System Synthesis (CODES+ISSS)*, 2011.

[21] D. Goswami, R. Schneider, and S. Chakraborty, "Re-engineering cyber-physical control applications for hybrid communication protocols," in *Design, Automation and Test in Europe (DATE)*, 2011.

[22] ——, "Co-design of cyber-physical systems via controllers with flexible delay constraints," in *16th Asia South Pacific Design Automation Conference (ASP-DAC)*, 2011.

[23] D. Roy *et al.*, "Tool integration for automated synthesis of distributed embedded controllers," *ACM Trans. Cyber Phys. Syst.*, vol. 6, no. 1, pp. 3:1–3:31, 2022.

[24] L. Zhang *et al.*, "Task- and network-level schedule co-synthesis of Ethernet-based time-triggered systems," in *19th Asia and South Pacific Design Automation Conference (ASP-DAC)*, 2014.

[25] D. Roy *et al.*, "Multi-objective co-optimization of flexray-based distributed control systems," in *IEEE Real-Time and Embedded Technology and Applications Symposium (RTAS)*, 2016.

[26] G. Tibba *et al.*, "Testing automotive embedded systems under X-in-the-loop setups," in *35th International Conference on Computer-Aided Design (ICCAD)*, 2016.

[27] J. Oetjens *et al.*, "Safety evaluation of automotive electronics using virtual prototypes: State of the art and research challenges," in *The 51st Annual Design Automation Conference (DAC)*. ACM, 2014, pp. 113:1–113:6.

[28] C. Hobbs *et al.*, "Safety analysis of embedded controllers under implementation platform timing uncertainties," *IEEE Trans. Comput. Aided Des. Integr. Circuits Syst.*, vol. 41, no. 11, pp. 4016–4027, 2022.

[29] M. Maggio *et al.*, "Control-system stability under consecutive deadline misses constraints," in *32nd Euromicro Conference on Real-Time Systems (ECRTS)*, 2020.

[30] D. Liberzon, *Switching in Systems and Control*, ser. Systems & Control: Foundations & Applications. Birkhäuser, 2003.

[31] S. Xu *et al.*, "Safety-aware flexible schedule synthesis for cyber-physical systems using weakly-hard constraints," in *28th Asia and South Pacific Design Automation Conference (ASP-DAC)*, 2023.

[32] A. Yeolekar *et al.*, "Checking scheduling-induced violations of control safety properties," in *20th International Symposium on Automated Technology for Verification and Analysis (ATVA)*, ser. Lecture Notes in Computer Science, vol. 13505. Springer, 2022.

[33] B. Ghosh *et al.*, "Statistical verification of autonomous system controllers under timing uncertainties," *Real Time Syst.*, vol. 60, no. 1, pp. 108–149, 2024.

[34] L. Ju *et al.*, "Context-sensitive timing analysis of Esterel programs," in *46th Design Automation Conference (DAC)*, 2009.

[35] ——, "Timing analysis of Esterel programs on general-purpose multiprocessors," in *47th Design Automation Conference (DAC)*, 2010.

[36] C. Hobbs *et al.*, "Quantitative safety-driven co-synthesis of cyber-physical system implementations," in *15th ACM/IEEE International Conference on Cyber-Physical Systems (ICCPS)*, 2024.

[37] H. Liang *et al.*, "Security-driven codesign with weakly-hard constraints for real-time embedded systems," in *37th IEEE International Conference on Computer Design (ICCD)*, 2019.

[38] P. Mundhenk *et al.*, "Security in automotive networks: Lightweight authentication and authorization," *ACM Trans. Design Autom. Electr. Syst.*, vol. 22, no. 2, pp. 25:1–25:27, 2017.

[39] S. Xu *et al.*, "Neural architecture sizing for autonomous systems," in *15th ACM/IEEE International Conference on Cyber-Physical Systems (ICCPS)*, 2024.

[40] Z. Wang *et al.*, "Bounding perception neural network uncertainty for safe control of autonomous systems," in *Design, Automation & Test in Europe Conference & Exhibition (DATE)*, 2021.

[41] M. Goyal, M. Dewaskar, and P. S. Duggirala, "Nexg: Provable and guided state-space exploration of neural network control systems using sensitivity approximation," *IEEE Trans. Comput. Aided Des. Integr. Circuits Syst.*, vol. 41, no. 11, pp. 4265–4276, 2022.

[42] V. K. Kukkala, J. Tunnell, S. Pasricha, and T. Bradley, "Advanced driver-assistance systems: A path toward autonomous vehicles," *IEEE Consumer Electronics Magazine*, vol. 7, no. 5, pp. 18–25, 2018.

[43] V. K. Kukkala, S. Pasricha, and T. Bradley, "Jams: Jitter-aware message scheduling for flexray automotive networks," in *Proceedings of the Eleventh IEEE/ACM International Symposium on Networks-on-Chip*, 2017, pp. 1–7.

[44] ——, "Jams-sg: A framework for jitter-aware message scheduling for time-triggered automotive networks," *ACM Transactions on Design Automation of Electronic Systems (TODAES)*, vol. 24, no. 6, pp. 1–31, 2019.

[45] ——, "Sedan: Security-aware design of time-critical automotive networks," *IEEE Transactions on Vehicular Technology*, vol. 69, no. 8, pp. 9017–9030, 2020.

[46] V. K. Kukkala and S. Pasricha, *Machine Learning and Optimization Techniques for Automotive Cyber-Physical Systems*. Springer Nature, 2023.

[47] J. Dey and S. Pasricha, "Co-optimizing sensing and deep machine learning in automotive cyber-physical systems," in *2022 25th Euromicro Conference on Digital System Design (DSD)*. IEEE, 2022, pp. 308–315.

[48] J. Dey, W. Taylor, and S. Pasricha, "Vespa: A framework for optimizing heterogeneous sensor placement and orientation for autonomous vehicles," *IEEE Consumer Electronics Magazine*, vol. 10, no. 2, pp. 16–26, 2020.

[49] J. Dey and S. Pasricha, "Robust perception architecture design for automotive cyber-physical systems," in *2022 IEEE Computer Society Annual Symposium on VLSI (ISVLSI)*. IEEE, 2022, pp. 241–246.

[50] A. Balasubramaniam, F. Sunny, and S. Pasricha, "R-toss: A framework for real-time object detection using semi-structured pruning," in *2023 60th ACM/IEEE Design Automation Conference (DAC)*. IEEE, 2023, pp. 1–6.

[51] V. K. Kukkala, S. V. Thiruloga, and S. Pasricha, "Roadmap for cybersecurity in autonomous vehicles," *IEEE Consumer Electronics Magazine*, vol. 11, no. 6, pp. 13–23, 2022.

[52] ——, "Indra: Intrusion detection using recurrent autoencoders in automotive embedded systems," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 39, no. 11, pp. 3698–3710, 2020.

[53] ——, "Latte: L stm self-att ention based anomaly detection in e mbedded automotive platforms," *ACM Transactions on Embedded Computing Systems (TECS)*, vol. 20, no. 5s, pp. 1–23, 2021.

[54] S. V. Thiruloga, V. K. Kukkala, and S. Pasricha, "Tenet: Temporal cnn with attention for anomaly detection in automotive cyber-physical systems," in *2022 27th Asia and South Pacific design automation conference (ASP-DAC)*. IEEE, 2022, pp. 326–331.

[55] H. Chishiro, K. Suito, T. Ito, S. Maeda, T. Azumi, K. Funaoka, and S. Kato, "Towards heterogeneous computing platforms for autonomous driving," in *2019 IEEE International Conference on Embedded Software and Systems (ICESS)*. IEEE, 2019, pp. 1–8.

[56] J. Wang, X. Wang, Y. Jiang, A. K. Singh, L. Huang, and M. Yang, "On evaluation of on-chip thermal covert channel attacks," in *2022 International Conference on Compilers, Architecture, and Synthesis for Embedded Systems (CASES)*. IEEE, 2022, pp. 9–10.

[57] Z. Long, X. Wang, Y. Jiang, G. Cui, L. Zhang, and T. Mak, "Improving the efficiency of thermal covert channels in multi-/many-core systems," in *2018 Design, Automation & Test in Europe Conference & Exhibition (DATE)*. IEEE, 2018, pp. 1459–1464.

[58] P. Rahimi, A. K. Singh, and X. Wang, "Fan speed control based defence for thermal covert channel attacks in multi-core systems," in *2022 29th IEEE International Conference on Electronics, Circuits and Systems (ICECS)*. IEEE, 2022, pp. 1–4.

[59] J. Xu, X. Wang, Y. Jiang, A. K. Singh, C. Gu, L. Huang, M. Yang, and S. Li, "Secured data transmission over insecure networks-on-chip by modulating inter-packet delays," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 41, no. 11, pp. 4313–4324, 2022.

[60] X. Wang, H. Huang, R. Chen, Y. Jiang, A. K. Singh, M. Yang, and L. Huang, "Detection of thermal covert channel attacks based on classification of components of the thermal signal features," *IEEE Transactions on Computers*, vol. 72, no. 4, pp. 971–983, 2022.

[61] X. Wang, S. Wang, Y. Jiang, A. K. Singh, M. Yang, and L. Huang, "Combating stealthy thermal covert channel attack with its thermal signal transmitted in direct sequence spread spectrum," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 41, no. 11, pp. 4064–4075, 2022.

[62] N. Navet, Y. Song, F. Simonot-Lion, and C. Wilwert, "Trends in automotive communication systems," *Proceedings of the IEEE*, vol. 93, no. 6, pp. 1204–1223, 2005.

[63] H. J. Jo and W. Choi, "A survey of attacks on controller area networks and corresponding countermeasures," *IEEE Transactions on Intelligent Transportation Systems*, vol. 23, no. 7, pp. 6123–6141, 2021.

[64] J. Laufenberg, T. Kropf, and O. Bringmann, "A framework for can communication and attack simulation," in *2022 IEEE 95th Vehicular Technology Conference:(VTC2022-Spring)*. IEEE, 2022, pp. 1–7.

[65] S. Vanderhallen, J. Van Bulck, F. Piessens, and J. T. Mühlberg, "Robust authentication for automotive control networks through covert channels," *Computer Networks*, vol. 193, p. 108079, 2021.

[66] X. Ying, G. Bernieri, M. Conti, L. Bushnell, and R. Poovendran, "Covert channel-based transmitter authentication in controller area networks," *IEEE Transactions on Dependable and Secure Computing*, vol. 19, no. 4, pp. 2665–2679, 2021.

[67] H. M. Song, J. Woo, and H. K. Kim, "In-vehicle network intrusion detection using deep convolutional neural network," *Vehicular Communications*, vol. 21, p. 100198, 2020.

[68] S. Zhuo, N. Li, and K. Ren, "Histcan: A real-time can ids with enhanced historical traffic learning capability."