Technical White Paper

Advanced Driver Assistant System

Threats, Requirements, Security Solutions

EXECUTIVE SUMMARY

This paper discusses security vulnerabilities and potential solutions for Advanced Driver Assistance Systems (ADAS). We introduce ADAS system architecture and present use cases. We further provide detailed threat analysis of two leading ADAS use cases: (1) lane departure warning and (2) adaptive cruise control. Based on threat analysis results we identify security problem areas and state security requirements for each. We devote the last part of this paper to ADAS security solutions that can meet identified objectives.

This study makes several key contributions to addressing ADAS security problems

- Establish critical needs to addressing security problems via detailed threat analysis
- Define main security problem areas for ADAS
- Identify challenges and requirements for securing ADAS control functions
- Establish the mission of securing E2E ADAS data path
- Define trust foundation for secure ADAS platforms
- Make recommendations for "ADAS security solution menu"

1. Introduction

Demand for Advanced Driver Assistance Systems (ADAS) is caused by desire to build safer vehicles and roads in order to reduce the number of road fatalities and by legislation in the leading countries. ADAS is made of the following physical sensors: radar, LIDAR, ultrasonic, photonic mixer device (PMD), cameras, and night-vision devices—that allow a vehicle to monitor near and far fields in every direction and of evolving and improving sensor fusion algorithms that ensure vehicle, driver, passenger's, and pedestrian's safety based on factors such as traffic, weather, dangerous conditions, etc. Modern ADAS systems act in real time via warnings to the driver or by actuation of the control systems directly and are precursors to the autonomous vehicles of the future.

Lead Author: Meiyuan Zhao
Research Scientist
Security & Privacy Research, Intel Labs
meiyuan.zhao@intel.com

Table of Contents

There are several challenges to design, implement, deploy, and operate ADAS. The system is expected to gather accurate input, be fast in processing data, accurately predict context, and react in real time. And it is required to be robust, reliable, and have low error rates. There has been significant amount of effort and research in the industry to solve all these challenges and to develop the technology that will make ADAS and autonomous driving a reality.

In addition to functional requirements, ADAS must be secured from adversaries with malicious intent whose goal is to compromise the system and cause catastrophic accidents with loss of life and damage to property.

It has been shown both in academia and automotive industry that control system can be compromised via malicious attacks launched through various means, for example via DVD player, the ODB-II port,[1,2] or even wirelessly via tire pressure sensors,[3] as a result displaying to the driver wrong warnings[3] or even causing fatality by remotely disabling braking system on a vehicle while it is moving.[1,2] In addition to protecting the system from criminal actors, there is a bigger threat looming from nation-state sponsored cyber terrorism.

In this whitepaper we argue that ADAS security should be considered as a fundamental non-functional requirement—together with reliability, robustness, performance, and low error rates. We analyze vulnerabilities in a conceptual ADAS architecture via representative use cases. Based on the vulnerability analysis results we state security requirements and make suggestions on countermeasures against malicious attacks. We show that ignoring ADAS security compromises other design goals.

2. ADAS System Background

ADAS system provides assistance to the driver and improves driving experience. Its primary function is to ensure safety of the vehicle, the driver, and the pedestrians or bikers. ADAS could be used to save fuel costs by enabling platooning in which vehicles follow each other within close distance; it could warn when a vehicle swerves across the lane or it could apply emergency brake to avoid collision, etc. To function reliably, ADAS must be able to recognize objects, signs, road surface, and moving objects on the road and to make decisions whether to warn or act on behalf of a driver.

2.1. ADAS Example Usage Cases

ADAS system is considered as the advancement from driver assistant system (DAS). DAS is a system that informs and warns, provides feedback on actions, increases comfort, and reduces workload by actively stabilizing or maneuvering the vehicle. ADAS system is considered as a subset of DASs, with increased use of complex processing algorithms to detect and evaluate the vehicle environment based on data collected via a variety of sensor inputs. Figure 1 demonstrates the spectrum of DAS capabilities available in production today; the capabilities considered as ADAS are highlighted with stars. The ADAS usage cases that require full power of real-time processing and intelligence are highlighted with full stars, whereas half-colored star marked usage cases are relatively more rudimentary ADAS cases.

2.2. ADAS Conceptual Architecture

To support ADAS functions the architecture must include modules for sensing, processing, intelligence generation, and decision making. Figure 2 is a generic view of what the ADAS system might look like. The overall system compromises sensors of various types; a CPU-GPU combination to perform the sensor data processing, object identification, and early sensor fusion; a "Central Brain" CPU for performing sensor fusion from different sensor blocks, object tracking, vehicle control activities to interact with the actuation, and a diagnostics block.

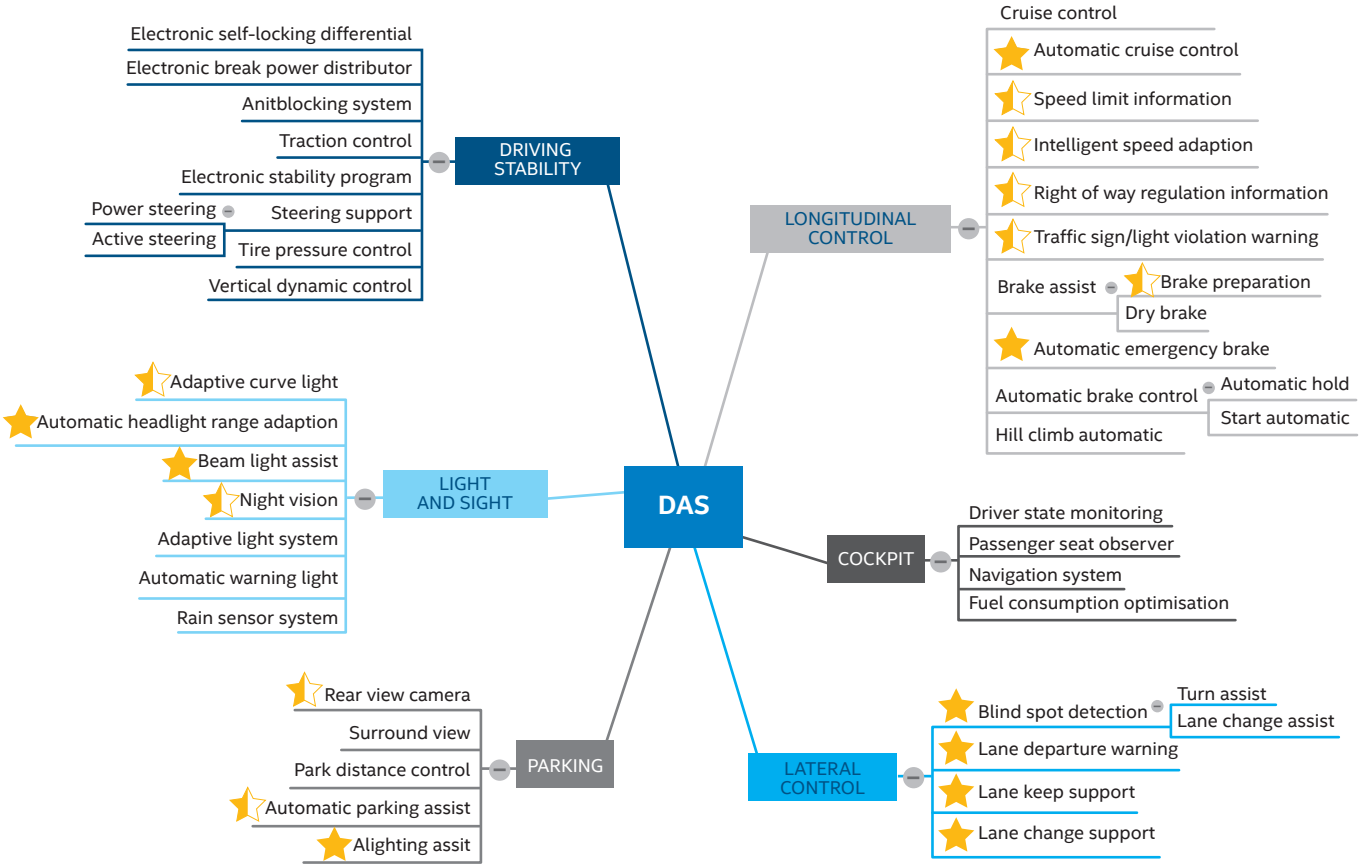

Figure 1. Spectrum of DAS and ADAS Functions

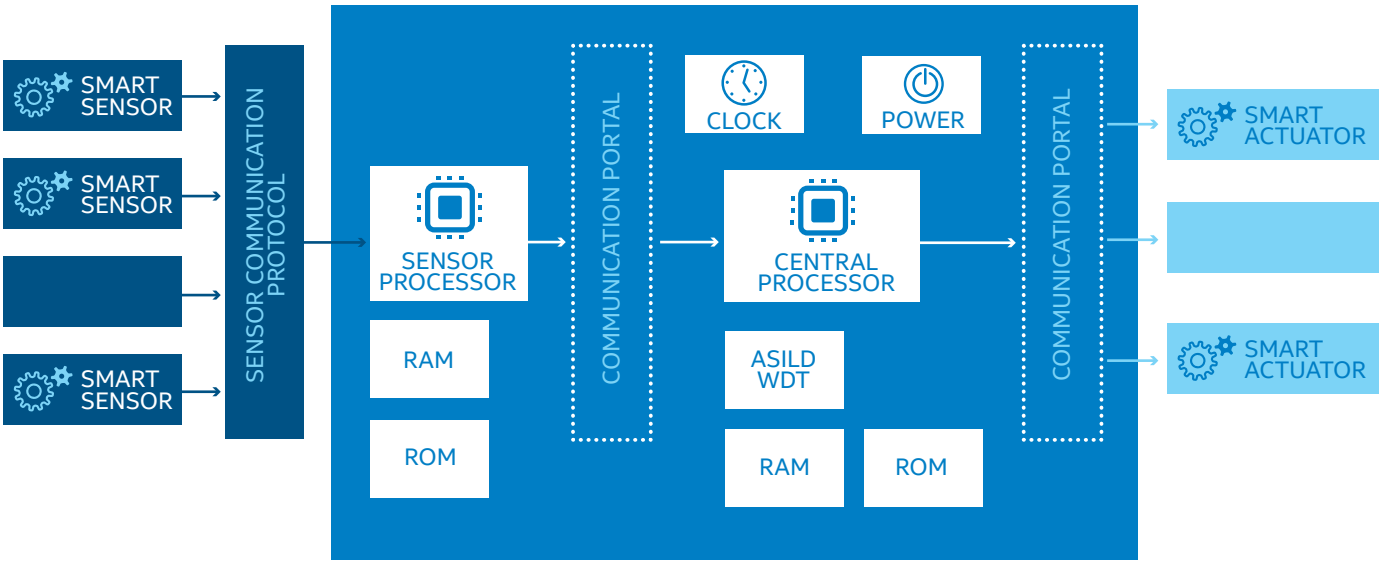

Figure 2. Conceptual Hardware Block Diagram for ADAS System

This system is considered as a close-loop control system, where the vehicle control actuation actions are computed based on received data from sensors. And the outcome of the ADAS actuation actions is fed back in the loop as sensor input. All the computing units in ADAS of the vehicular system are generally referred to as electronic control units (ECUs). The sensing and actuation ECUs are relatively resource constrained units, compared with the central processor of ADAS.

One of the key advancements in ADAS design is the concept of "sensor fusion." This is the process by which the internal processing takes input from the multiplicity of external sensors and creates a map of possible impediments around the vehicle. The map then facilitates the computation that creates a series of possible actions and reactions through situational analysis. Figure 3 shows an example ADAS-enabled vehicle with a collection of sensors to enable sensor fusion and actions.

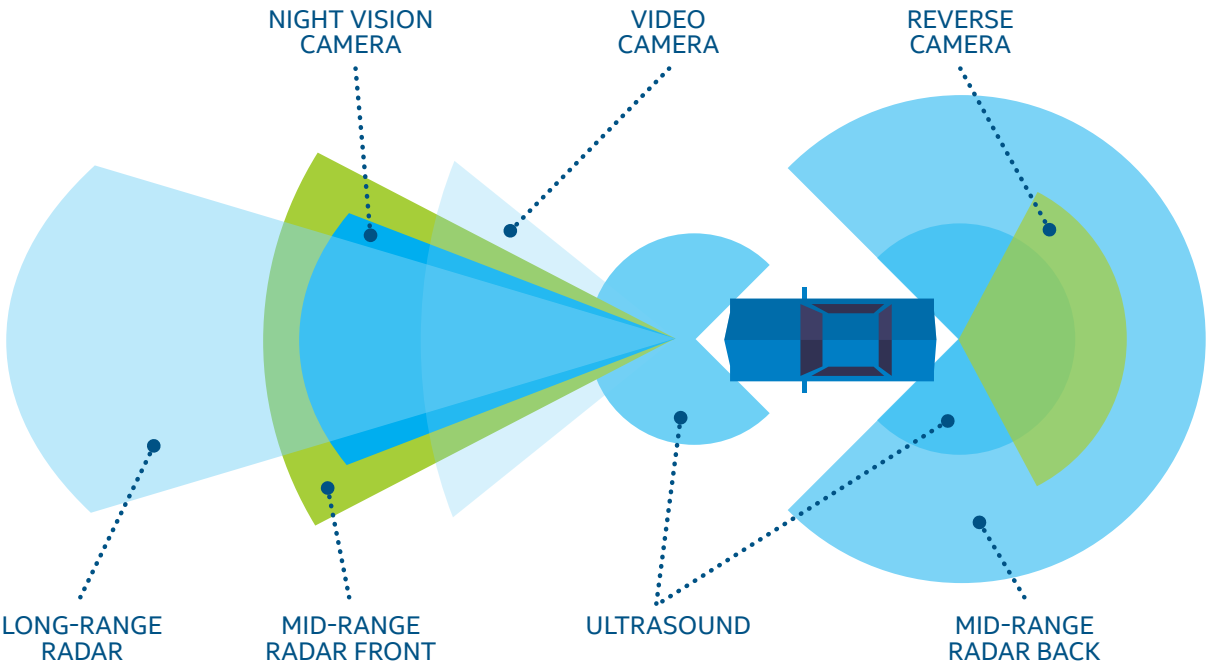

Figure 3. Example ADAS Sensors

Sensor fusion and situational analysis can be done case by case for different ADAS functions and occurs at multiple levels. It is beneficial to have early fusion (determining conditions as early as possible) and a centralized processing "brain" (improving quality of detection and reducing CPU power consumption). Figure 4 demonstrates an approach where sensor fusion occurs in both the sensor processor and the central brain. With this design, the system provides a horizontal architecture that can support multiple ADAS applications in parallel. Such architecture is an advancement from vertical systems that only support individual ADAS applications case by case. We base our security analysis on this conceptual architecture.

3. ADAS Security Problem Areas

Before looking into details of security threats, let us first examine, at high level, what are the major areas of concerns for ADAS system in dealing with hostile running environment and malicious actions by adversaries. In general, any malicious actions that could cause ADAS system to behave outside its specification are referred to as threats to ADAS. And the interfaces that allow such threats to occur are referred to as attack surfaces. Now the key questions are: what is the specified behavior of an ADAS system, and how do attackers cause the system to misbehave? The answers to these questions lead to the discovery of three major ADAS security problem areas.

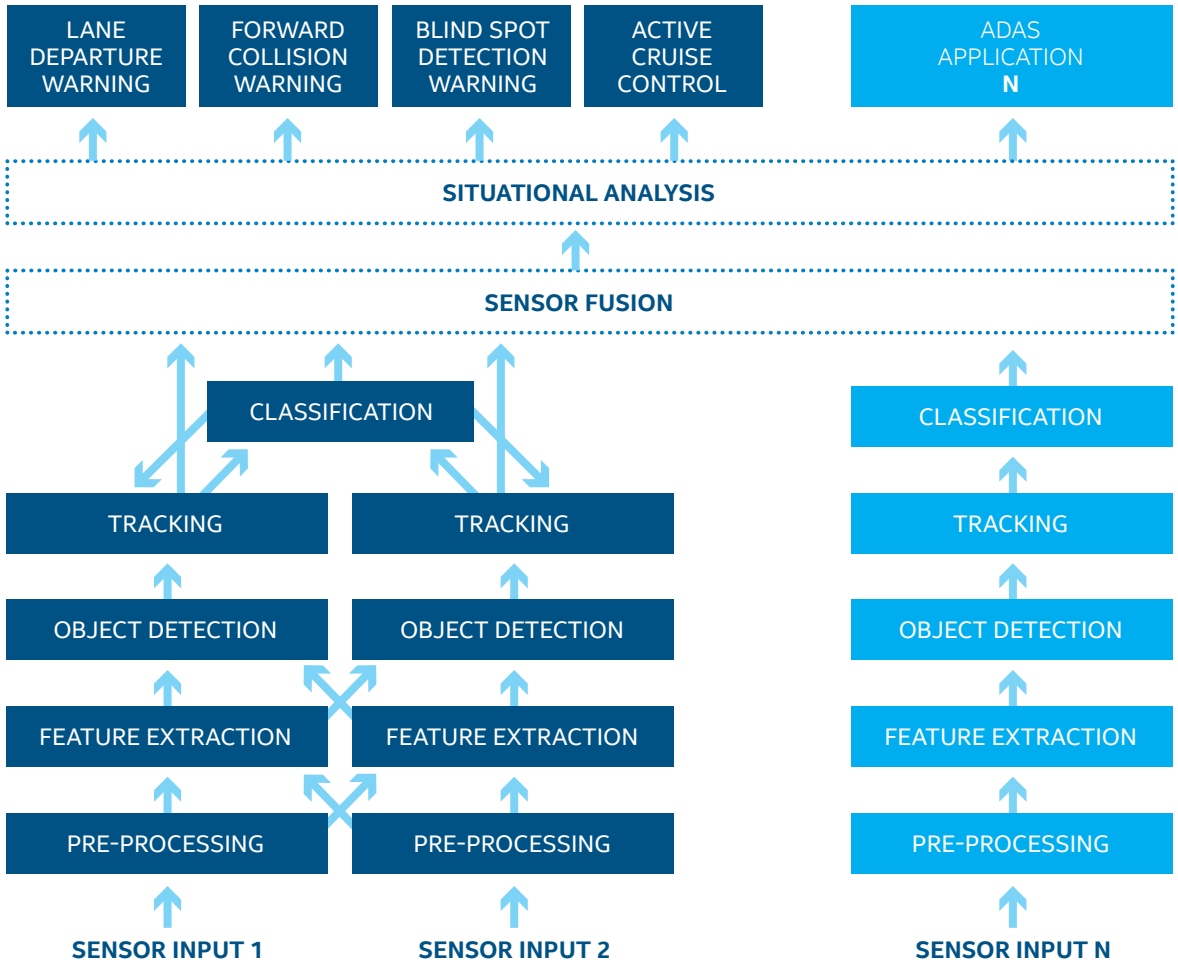

Figure 4. Example Sensor Fusion in ADAS

Control System Security

As we discussed before, ADAS can be thought of as a close-loop control system. While in operation it must satisfy functional safety, efficiency, performance, and reliability requirements. We refer to this whole system behavior as "ADAS control system and processing" and make securing it our priority.

Threats to the control function come from any actions the attacker could take, given their capabilities, to cause the system to act outside of its specifications. Any changes of the system properties that contribute to the violation of its safety goals may happen due to deliberate attacks. The security requirements to support safety goals should mainly concern establishing and maintaining functional integrity and other requirements. With further threat analysis and risk assessment, one could derive detailed security requirements, which will be discussed in Section 5.1.

ADAS Data Protection

In addition to attacking core functions of ADAS, the attacker could be motivated to attack the system to achieve other unexpected consequences by the original design. For example, the attacker could eavesdrop on ADAS data processing and/or internal communication to gain access to ADAS data. Leakage of data to an external party other than for local control system consumption may also be an unexpected behavior of the system, therefore a second area of security problem for ADAS may come from ADAS data protection. Security system must ensure Confidentiality, Integrity, and Availability (CIA) of data collection.

The practice of recording data for accountability may be implemented by a "blackbox" system. Should the blackbox be implemented, the storage security would be an issue. Similarly, integrity protected storage system would be required if the storage is used for collecting and storing other vehicle related information, such as object classifiers or maps. The specific threats are similar as seen in a typical storage system in the traditional computing world. Hence, in this whitepaper, we do not dedicate our focus on understanding threats in such systems, as well as the derived requirements on ADAS data protection. Nonetheless, there exist many studies in the literature that we could leverage. In this article, we derive security requirements based on existing threat studies and certain solutions can be leveraged to address security issues for data protection and access control.

Secure Lifecycle Management

Deploying and maintaining intended modules in the ADAS is as important as any other protection mechanism for ensuring ADAS system behavior according to the specifications. This process is typically referred to as lifecycle management. Changes to the ADAS system could be triggered by:

- System upgrade/algorithm updates
- Software patch
- Installation of new components for additional functions
- Hardware recovery and replacement
- System recovery due to compromise
- Root of trust update due to authority updates in the administrative domain
- Cryptographic algorithm and key updates due to cryptosystem migration or other reasons

ADAS is especially vulnerable to malicious actions during updates because some interfaces which are not normally available become open to external data or external operations. Furthermore, ADAS system consists of multiple modules. Any changes to any one of the modules will require the system to re-establish trust relationships between these modules so that they can reliably exchange data and commands.

Lifecycle management security is not a new problem for ADAS system. Any computing system needs to deal with changes to ensure that the system can "start secure—run secure—stay secure." The same problem in ADAS system is facing extra challenges:

- **Secure update on control system immature in auto industry and ecosystem**
  Manual update at a garage or repair shop by trained professionals is a common practice. Update process usually requires proprietary tools and labor intensive work. Although there is a new trend of attempting to enable remote update via standardized processes to relief the labor cost the procedure is still not mature enough to be pervasive. This is especially true for the updates that require intense verification on control system integrity.

- **Small scale ECUs to meet security primitive requirements for secure update**
  Updating relatively large scale computing platforms is straightforward since there are available system update and recovery technologies and services suitable for such platforms. ADAS, however, may have smaller scale micro-controllers for sensors and actuators. Such small scale platforms may not have sufficient cryptographic or security primitive support. Hence, the challenge is to achieve the same security objectives with lightweight system update security technology.
- **Long lifetime vs. limited cryptographic strength**
  Control systems like the vehicles typically have long life-times. Cryptographic solutions in the current computing world, however, have relatively shorter lifetimes. Hence, during the ADAS system lifecycle, there may appear the need to update the ADAS system with stronger and new cryptosystem. This problem in the traditional computing system is not critical given that most of the devices must be operational only for a few years. Careful design and analysis is required for updating the cryptographic system, because it effectively serves as the basis of trust for every security function. Compromising the root of trust will surrender control to attackers.

4. ADAS Control Function Threat Analysis

For conducting threat analysis, we need information on 1) target usage case; 2) architecture for the use case; 3) expected adversarial model. The analysis methodology is a commonly used approach where we decompose the system to assets and examine every interface exposed by each asset to understand all possible behaviors with all possible interface parameters and values.

4.1. Adversarial Model

Let's first define expected adversarial model in our analysis. We use a typical adversarial model, the same for analyzing threats to Intel's system product and technology. ADAS systems share many properties as a typical computing system. Hence, the adversarial model for the computing system is mostly applicable to ADAS system. We assume that ADAS system should worry about the attackers who can launch simple hardware attacks with the capability of "university challenge." This means:

- Expected attacker capability: Simple hardware attacker
  - Has reverse engineered all firmware, software
  - Can modify and replace all firmware, software
  - Can replace/substitute any ADAS components
  - Can remotely install privileged and unprivileged malware onto any micro-processor that communicates through external interfaces
  - Can read/write/jam/forge the radio channel
  - Can add/remove functionality
  - Can boot/operate removed parts in alternate environments
- Expected attacker capability: University Challenge
  - Will invest up to 6 months engineering effort and $50K part/equipment/computation to develop tools to attack many vehicles

Compared with the typical computing system, the difference in ADAS system is mostly on the physical aspect, besides the cyber actions and capabilities. For instance, the ADAS function used for controlling operation of the vehicle physically offers opportunities to attackers to launch their actions that may lead to consequences on control systems, on actuators, and on other ECUs that have mechanical impact to the vehicle. On the other hand, the attacker could also potentially launch attack via physical actions and eventually cause cyber or physical consequences. In our analysis, we attempt to cover both cyber and physical aspects of possible attacking actions.

4.2. Use Cases for Threat Analysis

Two applications are used in the study:

- Lane Departure Warning (LDW)
- Adaptive Cruise Control (ACC)

LDW use case is for vehicle lateral control, where warning is presented to driver if the ADAS system detects that the vehicle is departing from the current lane. The main functionality by the sensors and data fusion is to recognize the lane lines and predict the vehicle's driving direction based on the detected trajectory.

ACC use case is for vehicle longitudinal control. It manages the vehicle speed adaptively based on the detection of distance with leading vehicle, the current speed, the road condition, and prediction of leading vehicle's speed change. In this application, the ADAS system continuously generates actuation command to control throttle ECU or brake ECU accordingly.

These two applications use similar ADAS architecture, with some differences on required input sensing data and output data format and purpose. Their output difference is clear: warning only or take direct actuation. Potential failure in computing and generating corresponding output in these two cases may have dramatically different consequences.

These two use cases can represent several ADAS functions that improve driving experience and support safety. Our future work will extend the threat analysis on other use cases that facilitate parking or improving lighting and sight, as illustrated in Figure 1.

4.3. Summary of Threat Analysis Results

Detailed threat analysis is documented in Appendix A and Appendix B. Here, we provide a summary of threat analysis results and insight on high risks.

Study summary: entire data path from initial sensing to final actuation is vulnerable.

Let's examine where the vulnerable assets and exposed interfaces are. Main attack surfaces, shown in Figure 5, include exposed interfaces and are broken by vulnerable assets (internal or external). Attackers have a range of options, they can, for example, generate false data on a sensing platform, modify data on the internal communication channel, generate undesirable ADAS output data on the fusion brain platform, change output data on the internal communication to the actuation ECUs, manipulate firmware and software on the output platform to make the system fail.

Our analysis reveals that there is a set of data properties especially vulnerable to manipulation. Beyond the basic attack on data values, as summarized in Figure 5, vulnerabilities exist if syntax, semantics, timing, availability, and correlation are manipulated. Data syntax refers to some of the properties associated with the content, for example, output from LDW could be played via audio device where volume of the output warning is defined as data syntax. Change of audio volume to undesirable level is a realistic threat to LDW function.

Given the diverse types of data being generated and processed in the ADAS, not only forged data or incorrect data content has impact to the system, but also whether the data is in time for consumption, or available at all are intrinsically important issues for ADAS control system. Missing data or delayed data may cause the system fail to generate appropriate intelligence, or respond to situations in real time. Furthermore, the sensing fusion algorithms rely on potentially multiple streams of data. Some algorithms have requirement that streams of input data are received and processed in correlated sequence. Hence, attack actions that can successfully change the correlation to further manipulate the behavior of sensing fusion.

In terms of undesirable consequences, as summarized in Figure 5, the compromised ADAS system could cause false positive or false negative warnings or actuation actions. In false positive case, the ADAS system could generate warning or take unnecessary actions on vehicle control system to respond to falsely computed "need to warn or take action" situation, whereas the actual driving condition may be still normal. On the other hand, the false negative outcome could cause the system fail to respond to potential danger happening on the road. Furthermore, the control decision could still be relevant, but only relevant to the past condition, a little too late current. All these undesirable outcomes could potentially lead to unsafe driving consequences, cause a collision, or even loss of human life. These consequences

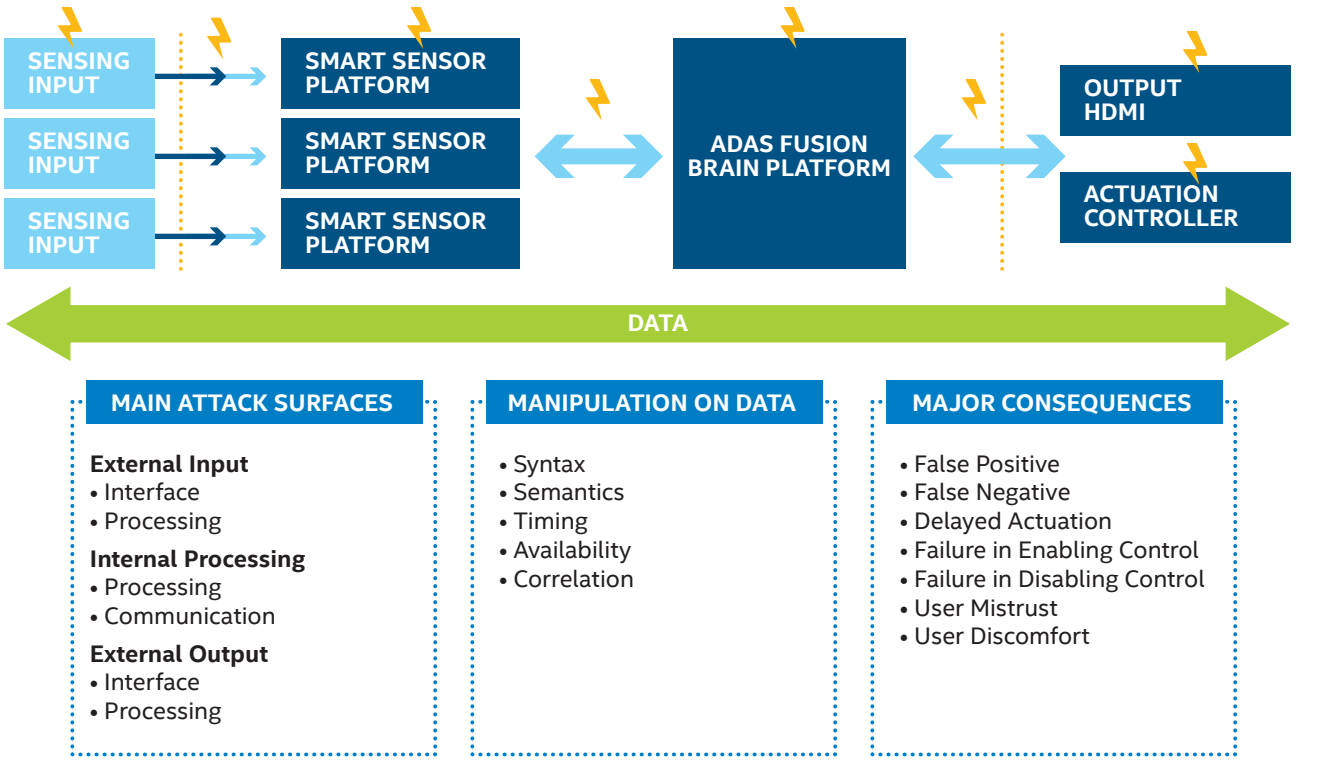

Figure 5. Summary of Threat Analysis Results

are very specific to the vehicle control system and defeat its primary design goal: improving driving safety.

Other consequences may include failure in enabling or disabling ADAS control. In addition, frequent failures in reporting accurate driving conditions can cause user's mistrust of the ADAS system and turn off the system all together.

ADAS system can also be compromised to defeat the goal of improving driving experience. For instance, in the case of ACC, to maintain a proper speed, the vehicle may be manipulated to speed up or slow down suddenly, or repeatedly take these actions. Such outcome is apparently not desirable to the driver and to passengers.

5. ADAS System Security Requirements

Given the threat analysis results and insights in highly vulnerable parts of the system, we examine security and system requirements that will provide guidance in designing security solution for ADAS.

5.1. Control System Security Requirements

Security requirements for ADAS control system are primarily derived from threat analysis results. We take LDW and ACC as case studies, and focus most efforts on understanding how attackers could launch the attacks that could cause the system to act outside its control system specification. That is, the system that achieves the control system objectives.

Therefore, the **overall security objective** on ADAS control system is to:

> Defend the ADAS control function against malicious attacks, so that the ADAS control function can achieve the expected specified behaviors: delivers warning or takes necessary actuation actions in real time that accurately reflect the current driving condition and according to reasonably accurate prediction of potential danger.

To support this objective, the secure ADAS control system should satisfy the following major requirements.

- **R1. Availability** The system and functions should ensure that data and processing capability are available to satisfy the needs by ADAS fusion and intelligent actions.
- **R2. Real Time** Delivering of warnings/actions should be in real time to be useful.
- **R3. Accuracy** Warnings and actions correctly and accurately reflect the current driving condition and accurate enough prediction of potential incidents on road.
- **R4: Reliability** System is able to predict potential dangerous conditions with high probability and low error rate; Ensuring such capability when system is under attack.

These requirements are fundamental for a secure ADAS control system. They shall be used guidelines when designing and implementing the ADAS system. In the complete ADAS data path, assets that satisfy these requirements include all types of data, sensing modules, actuation modules, any processing modules, and internal communication. Further decomposition is needed when it comes to the need to design solution to protect a specific module or a specific set of modules in the ADAS system.

On *sensing modules*, initial calibration and on-going operation are the focus. Sensing data is required to be available in real time, accurately reflect the sensing condition, and reliable to tolerate most of unexpected conditions.

On *actuation modules*, similarly, the focus is on initial calibration and on-going operation. In particular, it is required that incoming actuation commands are processed in real time and accurate manner. The actual actuator should execute the commands accurately, fast, and reliably under various conditions. The actuation function should be available and continuously in correct operation status.

On *internal processing* modules, software and hardware protection should be in place to ensure that the supporting data is 1) available; 2) not delayed or rushed; 3) content is authentic and integrity protected; 4) syntax is correct; and 5) correct correlation in multiple sensing streams is maintained. To accomplish these objectives, the major software and hardware components for any internal processing module should:

- Provide boot time system integrity protection against malicious software modification
- Provide run time execution protection against malicious modification
- Ensure processing latency protection against malicious system jamming or denial of service

On *internal communication*, to protect data properties, the protocols should ensure integrity, authenticity, availability, freshness, and timeliness of any data internally communicated. These requirements can be used to further derive specific requirements on functions for internal communication, including managing communication keys, handshakes, communication buffering, and actual data distribution.

5.2. Lifecycle Management Security Requirements

Like other secured systems, the functions and the architecture that support ADAS functions should start secure, run secure, and stay secure. Security from cradle to grave is a critical requirement to maintain ADAS lifecycle integrity. ADAS system consists of multiple components and the internal communication. This makes security of lifecycle management more challenging compared with a traditional computing system or any other systems that only deal with a single main processing module. In this section, we pay close attention on security management of ADAS ensemble and modules that specific for ADAS or automotive. Readers can refer to literature for secure lifecycle management requirements for stand-alone computing modules.

5.2.1. Start Secure

Primary goal at this stage is to establish secure provisioning of ADAS components, and the trust relationships between components. Below are some key requirements:

- **Establish Root of Trust (RoT):** A trusted owner of the entire system to be established and held accountable of ADAS system configuration and management. This common RoT is the foundation for ADAS components to establish secure relationships and communication keys as required by ADAS operation.
- **Deploy trustworthy HW and SW trust modules:** It is required that the initially provisioned system hardware and software modules are trustworthy and are the authentic version by the authority. The system is also required establish the authority as the trusted party, and provide mechanisms to allow third party to gain proof that the system is running trustworthy modules.
- **Establishing integrity policies among components:** There should exist policies and mechanisms that allow each individual component to establish and prove its trust status to other components, as well as for the components to establish and prove their trusted status as a whole architecture and group.

5.2.2. Run Secure

Once the ADAS system starts to run, the following are some high level requirements.

- **Invoke trusted system only:** Upon system boot, only the provisioned trusted hardware and software can be booted. Any forged components should be detected and prevent from loading into the system.
- **Validated input only:** Mechanisms should be invoked here to validate the input. In particular, there are interfaces exposed externally to acquire input from physical world. Mechanisms that are used to protect input at these interfaces are still in their infancy.
- **Detect runtime attacks:** It is required that system should at least detect attacks when they happen in the system. To do this, the system is required to generate information of system status that differentiates between "normal" conditions vs. "under attack" conditions.
- **Prevent runtime attacks:** A stronger requirement for system to "run secure," is to invoke mechanisms that prevent the system from being attacked at runtime. Solutions that satisfy this requirement may be built-in design in the system architecture, or additional security mechanisms/protocols that protect the operation and data.

5.2.3. Stay Secure

Changes occur in ADAS system with respect to individual components and managing the ADAS trusted ensemble. Whether it is about updating/patching hardware and software on individual components, or on updating trust relationship among group members, the following are the high level requirements to be satisfied in order to maintain the trusted ADAS ensemble.

- Integrity protected
- Authenticated source to provide update
- Update with authorized modules only
- Freshness in update process
- Proper trust foundation re-establishment on updated components
- Group-based authenticity and integrity protected

6. Secure End-to-End ADAS Data Path

This section offers some thoughts on how to design and develop solutions to support ADAS security requirements.

Given our analysis of threats and security requirements, the overall goal for securing ADAS function has emerged: ensuring protection of data collection, processing, and control system execution for ADAS use cases. We are in the mission of securing the end-to-end ADAS data path: from collection to final consumption. Security solutions should be designed to secure ADAS path with four major areas of concerns: 1) secure Real-time Sensing and Input, 2) internal data processing, fusion, and decision making, 3) trustworthy and reliable output, and 4) trusted internal data dissemination. Furthermore, lifecycle management issues will be addressed together with control system security in our solution discussion.

Note 1: The discussion on securing diagnosis and auditing functions that require separate architecture from main ADAS system and are considered out of scope for this document.

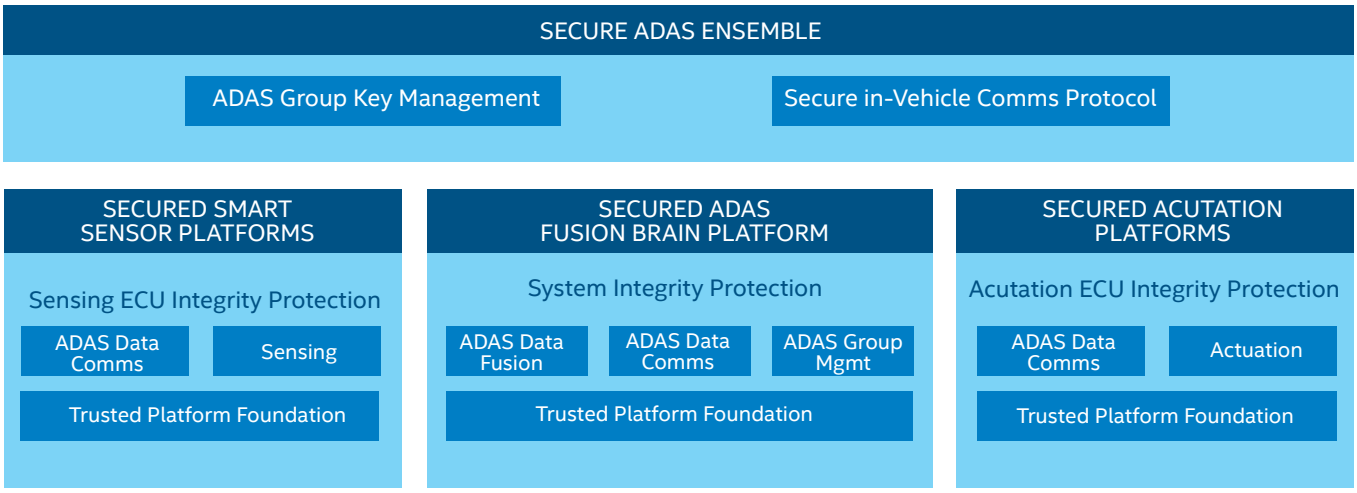

Figure 6. Summary of ADAS Security Solution Areas

Note 2: We don't necessarily limit ourselves in a specific solution space. As general discussion, all solution spaces—hardware, software, firmware, system, networking, and services—are under consideration.

6.1. Summary—Solution Areas

To secure the end-to-end ADAS data path, the main solution areas should concentrate on following five areas:

1. Common trust basis for computing platform
2. Securing sensing
3. Securing actuation
4. Securing internal processing
5. Securing ADAS ensemble (trust management and communication)

Their relationships can be described at high level, as illustrated in Figure 6.

To handle input, or output, or internal processing, any computing platform should have a set of basic security features that establish basic trust on these computing platforms. Beyond this basis, each computing platform may be required to enable additional security solutions that meet specific security and functional requirements for input, output, or internal processing respectively.

As illustrated in Figure 6, for sensing ECUs, the primary tasks for security solution are protected sensing and trustworthy communication to distribute sensing information to consuming components in the ADAS system. For actuators, the ADAS specific security protection is trustworthy actuation, and communication for receiving actuation commands and other management messages. For main ADAS main fusion platform, the center of the solution stack is on trusted data fusion processing.

Furthermore, the trust relationships between these platforms and how they establish secure communication to support security requirements on data flows are also critical. In this area, there are key management issues among platforms, and issues of securing data paths and channels. To facilitate trusted ensemble management, there should be a module to manage the ADAS group. Here, we hypothetically put this functionality on the central fusion platform, given its unique position that allows it communicate with every other component in the ADAS system. This function could also be enabled by a specially designed module as a completely separate component in the system.

In the rest of this section, we discuss each solution area in more detail. Again, we mainly focus on how each component in the system can leverage security technology to satisfy security requirements for ADAS control system and lifecycle management. In some of the areas, the existing security technology needs to be re-engineered or re-designed to meet ADAS requirements.

6.2. Secure ADAS Computing Platforms

Computing platforms in ADAS include sensing platform, the ADAS main processing platform, and the platforms for actuation. All these platforms should establish a set of primitives as trust foundation for supporting ADAS security operations. Figure 7 demonstrates these primitives at the conceptual level. Note that the actual instantiation of these primitives is case by case depending on the target platform or micro controller. Also, this is a demonstrative security profile. We do not require that every ADAS component to support all of these primitives. This is especially the case for resource limited micro controllers. Nonetheless, for the completeness of discussion, it is important that the trust foundation covers major areas of issues for protecting basic functions that an ADAS component relies on.

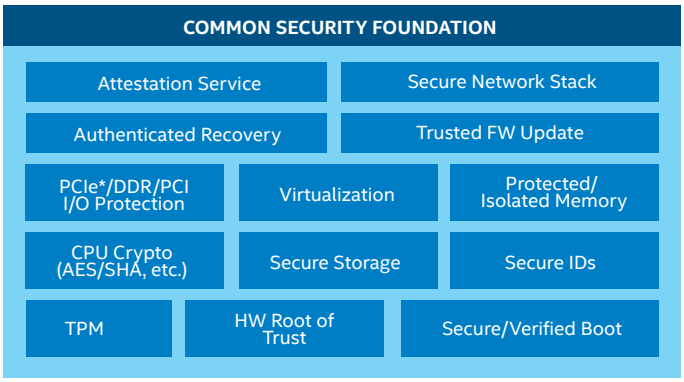

Figure 7. A Menu for Security Ingredients for ADAS Computing Platform

In summary, a platform is responsible for four major security goals in order to support ADAS functional security.

Goal #1: Establish and Maintain Platform Integrity

Goal #2: Support Secured Update

Goal #3: Protect ADAS Processing and Operations

Goal #4: Protect Data Communication

Establish and Maintain Platform Integrity
Platform integrity is the basis of trust for the platform. In this category, the platform should enable primitives to form a trusted computing base: trusted platform module (TPM) or equivalent component, HW root of trust, secure IDs, secured storage, protected crypto engine. Based on these primitives, other basic security functions can be established, including secure/verified boot, attestation of system and firmware.

Support Secured Update
The platform should have capability to continuously watch itself for integrity protection. In the case of corruption, on either BIOS, firmware, OS, or on software, the authenticated recovery technology should be invoked to ensure the system can regain its integrity with the authenticated image. Furthermore, the trusted FW/SW update technology is part of trust foundation to ensure system can be securely patched.

Protect ADAS Processing and Operations
Protection on ADAS function operations on the platform is needed to achieve security requirements R1—R4. The execution integrity is the first priority. Furthermore, internal data structure and buffered data should be protected from malicious tempering. For this purpose, the security foundation can provide primitives for boot time and run time execution protection. Example primitives could include secure/verified boot, secure storage, protection memory for execution isolation, and virtualization.

Protect Data Communications
To ensure strong protection on internal communication for ADAS data, one could enable technology to protect channels directly. This may include I/O protection on platform and system, as well as the secured network stack support. Furthermore, the security protocol should be in place to ensure that data transportation satisfies the requirements: authenticity, integrity, freshness, completeness, and timeliness. The communication driver should be operated in the protected execution environment with support of HW root of trust and secure identities for necessary cryptographic operations and validations. Also, the networking buffer operated by the driver should be in a protected memory region to prevent it from being malicious tampered.

6.3. Intel ADAS Platform Security Foundation

Intel is uniquely positioned to provide full hardware, firmware, and software support to ADAS trusted foundation. Intel continues to enhance systems so they run more securely. A key component of this approach is providing more robust, vulnerability-resistant platforms. Security features are embedded in the hardware of Intel® processors, some of which will be deployed and leveraged in future ADAS systems. The following is some sample technology that combines the hardware strength from Intel® platforms and comprehensive software security solutions from McAfee, offering great potential to enable strong ADAS E2E solutions.

Intel® Boot Guard for secure boot
Intel® architectures provide temper-resistant hardware modules, such as authenticated code module (ACM)-based secure boot that verifies a known and trusted BIOS is booting the platform. It is the foundation of trustworthy system boot and execution.

Intel® Platform Protection Technology with BIOS Guard
BIOS Guard supports hardware-assisted authentication and protect against BIOS recovery attacks, along with other platform protection technology, ensuring fundamental platform protection.

Intel® Platform Protection Technology with Platform Trust as basis for ADAS key management
Platform Trust provides integrated solution for secure credential storage and key management. This feature may be utilized for enabling secure ADAS ensemble key management and serving as the basis for trusted ADAS communication.

Intel® Trusted Execution Technology (Intel® TXT) enhancing platform security
To protect against attacks toward hypervisor and BIOS, firmware, and other pre-launch software components, Intel® Trusted Execution Technology (Intel® TXT) provides hardware-based technology to establish a root of trust through measurements when the hardware and pre-launch software components are in a known good state.

Hardware-Assisted Intel® Data Protection Technology
With Intel® AES New Instructions (Intel® AES-NI), Intel® Secure Key instruction, and Intel's Digital Random Number Generator (DRNG), users gain combined value of solid cryptographic foundation support from Intel platforms.

Intel® Virtualization Technology (Intel® VT) securing ADAS workloads
On shared virtualized hardware, a set of ADAS workloads can co-locate while maintaining full isolation from each other, freely migrate across infrastructures, and scale as needed. Intel® Virtualization Technology (Intel® VT) provides hardware assist to virtualization software, eliminating performance overheads, and improving security.

Intel® Software Guard Extensions (Intel® SGX) for targeted ADAS execution protection
With a new set of CPU instructions, Intel® Software Guard Extensions (Intel® SGX) could provide ADAS application capability to set aside private region of code and data, allowing application to protect sensitive data from unauthorized access or modification by rogue software, or enabling the platform to measure ADAS application's trusted code and produce a signed attestation.

TrustLite: Lightweight Trusted Execution Protection Technology[4] for small ECUs in ADAS
The technology developed in Intel Labs to enable protection on execution on very resource-constrained SoCs or platforms. Potential protection on every small ECUs (sensing, processing, or control) become possible with TrustLite,[4] enabling Secure Loader, Secure Exceptions/IPC, and Lightweight Isolation Execution.

McAfee Deep Defender and DeepSAFE defending malware attacks
Anti-malware solutions work with the Intel hardware features to run beyond the operating system to detect covert stealth attacks.

McAfee Embedded Control to protect fixed-function ADAS modules
Fixed-function modules, including ADAS fusion brain platform are protected against any unauthorized change on the application with McAfee Embedded Control technology.

Intel is thriving to develop platform protection foundation across various form factors, making E2E ADAS system protection possible. The rest of this section discuss specific additional treatments for secure sensing, actuation, data fusion, and ensemble management.

6.4. Secure Sensing

Sensors are at the starting line of the ADAS data path. The sensing input has critical impact on overall correct operation of ADAS functions. Leveraging the common trust foundation, one can ensure that the smart sensing platform satisfies the requirements and enables a trusted computing base for ADAS sensing function, and secured sensing data communication. For the platform that supports smart sensing, there are few issues that require special attention and treatment.

- **Lightweight Security Mitigation**
  The variety of sensing platforms demands that the security solutions should be customized to be suitable for the target platform resource constraints and cost limit. For some of the lightweight sensors, the full protection package as described in Section 6.2 is not feasible. Tradeoffs between cost and just enough security are needed.

  For small scale micro controllers, the trusted computing base may be as small as a protected memory region for sensing execution. To minimize the complexity, the system could be required to support only a static version of the sensing algorithm firmware. No other software or firmware can be inserted to the system without authorization. Such concept can be realized by an *embedded controller whitelisting* solution. Furthermore, the *cryptographic* support of small scale embedded controller may be simplified as well. The goal, is to support "just enough" for customized protection on the sensing platform.
- **Trusted Graphics Processing**
  Some of the sensing platforms used in ADAS require powerful graphics processing. For instance, the front-view and side-view cameras operate to capture frame streams at high frequency in real time and generate initial graphics models, and output to main ADAS processing platform. To meet the performance requirement, the platform may integrate a GPU to facilitate demanding graphics processing. In such architectures, the additional care of security is needed on GPU processing and data paths to ensure *authenticated data flows between GPU and CPU*, as well as the *trusted execution environment support on GPU*.
- **Sensing Integrity Protection**
  Smart sensing platforms interact with physical environment directly. Trusted ADAS operations rely on authentic, complete, and fresh source of information, which is the main task of sensors in ADAS. The challenging objective is to provide ***sensing integrity technology*** that can ensure the generated models by the sensing platform accurately and correctly reflect current physical condition in real time. This is, currently, a technology gap to be filled with further innovation.

6.5. Secure Actuation

The secure actuation platform is on the final consuming phase of ADAS data path. Similar to sensing platform, actuation ECUs could leverage the common trust foundation for target security requirements. For the platform that supports reliable actuation, there are few issues that require special attention and treatment. Some of the issues are similar ones that the smart sensing platform faces.

- **Lightweight Security Mitigation**
  Actuation platforms are typically small scale and support singular functionality. For these smaller scale controllers, the full protection package as described in Section 6.2 is not feasible. Tradeoffs between cost and just enough security are needed. Again, the minimized trusted computing base, as required for small sensors, is also required for small scale actuation platforms. Hence, ***lightweight cryptographic engine*** and ***embedded controller whitelisting*** solution are of special consideration to support lightweight security mitigation on actuation.
- **Actuation Integrity Protection**
  Actuator ECUs manipulate control mechanics of the vehicle directly. Taking authentic input command and making sure the execution is faithful and timely are critical to the overall ADAS system security and reliability. Any failure of actuator ECUs could have catastrophic and fatal consequences. While most actuator ECUs are implemented to satisfy highest safety requirements, we need to enable security protection on these ECUs so that possibility of triggering failures cannot increase when the ECU becomes the target of malicious attack.

  Here, the challenge of ensuring integrity protected actuation has not been fully met, which remains a technology gap to be filled with further innovation.

6.6. Secure ADAS Main Data Processing

As the main "brain" of ADAS system, the ADAS main processing platform demands full security protection. It is the best practice to enable strongest security protection on this platform given its critical function of data fusion. The good news is that the target platform is relatively more powerful than sensors and actuators. Therefore, a full set of security primitives can be made available on the main platform for integration to ADAS internal data processing protection. There are three particular considerations on main platform worth further discussion.

- **Maintain Functional Simplicity**
  As powerful as it should be for processing high volume of data in real time and generate accurate models from multiple data streams, the main ADAS computing platform in fact supports relatively less complex functionality: generate control commands to manage vehicle operation, given current driving condition. Unlike a typical computing platform in IT world (e.g., desktop, laptop, or smartphone), the ADAS computing platform only supports a few algorithms and a few ADAS use cases.

  With this observation, it is important for the ADAS main computing platform to maintain its simplicity. The security technology that ensures only a few authorized applications, such as whitelisting, can be invoked. And unauthorized installation or modification of the ADAS application software should be detected and denied.
- **Protecting Real time Operating System and Software**
  Huge work loads of ADAS main platform and the real-time reaction requirement could drive the system architecture to be supported by a real-time operation system. This means, the security ingredients that are traditionally tied to a specific operating system should be re-engineered or re-designed to be suitable for the real-time operating system and software. Example of such technology may include boot time/run time integrity protection, attestation, anti-malware solutions, and trusted recovery technology.
- **Multi-interface Protection and Isolation**
  The main ADAS computing platform is situated in the unique position in the vehicular control system that it has various interfaces that essentially connect to every parts of the ADAS system, from user interface, physical sensing, and to passing commands to ECUs. The more interfaces it has, the higher probability that the system may be attacked. Therefore, protecting each interface, as well as isolating these interfaces are critical to the system to have any hope of maintaining its integrity. On the main ADAS computing platform, an integrated isolation solution, such as ***"Trusted Execution Environment + Protected Data Path"*** should be carefully designed and implemented. Additional challenge in this task is that the resulting solution has to also meet the "minimized latency" requirement as mandated by the real-time system.

6.7. Secure Connected Ensembles

The security system for intra-vehicle data communication for ADAS function consists of three components:

- Integrity assurance of communicating modules
- Key management that establishes trust between modules
- Secure communication protocol

Integrity Verification

The integrity of platforms serves as the basis for establishing trusted channels between them. Here the ***local attestation or integrity verification solution*** is required to establish that the modules maintain their integrity of the system. The integrity demonstration to modules serves as the basis of trust for further establishing protected channels.

Furthermore, if there is a need that a group of modules to establish trusted communication, the basis of trust among them could be established via a ***group-based attestation*** solution. There are some technology in the literature, could be used as the basis for constructing group-based attestation. Yet, to our best knowledge, this is an open challenge for securing ADAS group.

Secure Communication Protocols

For data distribution, we need to enable trusted data paths across platforms in the ADAS architecture. As identified in threat analysis, the primarily concerned data channels are mostly bi-directional from the main ADAS processing platform to all other platforms including: smart sensing platforms, platform for user input, ECUs for status report and actuation control. The primary issues for protecting data flows are the data properties that directly impact functional goals of the entire ADAS control system: availability, real-time, reliability, and accuracy.

Supporting secure communication among ADAS modules is a non-trivial problem. As demonstrated in threat analysis, given the expected adversarial model, communication channels can be fully under control by adversaries. To protect data flows on these channels, security protocols are required. Security primitives such as HW root of trust, secure identities, and secured cryptographic engines are required to support cryptographic operations and validation by the secure communication protocol. In addition, the complexity of supporting secured communication comes from two issues: ***diverse channel types*** and ***diverse data flows***.

For security design, different layers of network stack have their own security issues and suitable solutions. In ADAS system, there exists diversity of communication channels: Ethernet, CAN, FlexRay, and other proprietary channels. Different type has its own security issues. And existing solutions are different in terms of the technology maturity. For instance, for Ethernet links, existing TCP/IP stack and their security countermeasures could be suitable to secure communication. On the other hand, the CAN protocol is currently still vulnerable to malicious attacks and the solution is still under development in the standards groups and by industrial consortia. Here, the main issue with enabling secure communication in ADAS system is primarily on ensuring ***interoperability***, so that modules from different vendors could be easily integrated to construct secure intra-system communication in an ADAS system.

The communication protocols may not be always pairwise between modules. Depending on the ADAS use case, there could also be group-based communication. For instance, in the case of multiple sensors serving together for ACC function in ADAS, there may be needs to ensure multiple front-view cameras work together to ensure reliable modeling of front view road condition. There may be group-based communication flow among these cameras, the corresponding sensing platform(s), and the main ADAS processing platform. That means security protocols should be designed and deployed among these modules that satisfy specific requirements by such group communication workloads. Hence, individual modules in ADAS system should provision and invoke these additional components that support data flow protection, whether it's pairwise based or group base, or any form of relationship.

Key Management

To support secure communication protocol, the secure communication sessions should be established. The goal of key management is to establish session keys or the keys used to protect the communicate channel between parties. In ADAS system in general, the communication is required to support data availability, freshness, integrity, and authenticity. Hence, the keys used to achieve these goals should be established properly before the modules could engage with the secure communication protocol for exchange data. The property of these keys depends on the choice of target security protocol.

The basis of establishing keys is from the trust relationship between the modules. The trust relationships should be represented by the security credentials that can authenticate the modules. In addition, the binding of the module's current status with its authentication credentials could be required to establish the trust. To achieve this goal, the system needs to utilize the integrity verification function. Such verification may be required to be mutual: mutual authentication and mutual attestation. Furthermore, the group-based authenticated communication requires group-based trust relationship, hence requires member authentication in group and group-based attestation.

Technology foundation in this area exists. The suitable solutions for ADAS should be designed to meet the architectural requirements and specific communication security requirements.

7. Conclusion

This paper details the security issues that directly impact the ADAS system functional safety goals. Key security issues are identified via threat analysis case studies. Security requirements are derived from every parts of the ADAS architecture. Key security objectives are defined to support ADAS control system integrity, data protection, and lifecycle management. Various pieces of the security architecture and solutions are identified that can be put together to protect the vehicle's operation with ADAS system. There are many architecture specific details to be worked out. More learning comes from actually doing experiments. Further work is along with a few specific directions:

1. Further exercise and experiment for a specific ADAS architecture; engineering work to enable ADAS security solutions, advancing the secure ADAS system to be reality.
2. Proof-of-concept and experiments:
   a. Integration of primary security technology with a target ADAS use case.
   b. Validate effectiveness of solutions for control functional safety.
   c. Evaluate performance overhead and impact on functional correctness.
3. R&D efforts to tackle challenges and technology gap in ADAS security, both from functional and performance perspectives.
4. Security impacts on functional safety for automotive: detailed analysis, and community consensus building in the context of automotive standardization.

References

1 Checkoway et al, "Comprehensive Experimental Analyses of Automotive Attack Surface," USENIX Security, Aug. 2011

2 Koscher et al, "Experimental Analysis of a Modern Automobile," S&P 2010

3 Rouf et al, "Security and Privacy Vulnerabilities of In-Car Wireless Networks," USENIX Security, Aug. 2011

4 Koeberl et al, "TrustLite: a security architecture for tiny embedded devices," In Proceedings of the Nineth European Conference on Computer Systems (EuroSys '14)

Appendix A. Threat Case Study #1: Lane Departure Warning (LDW)

A.1. Lane Departure Warning Usage Case

The lane departure warning (LDW) use case refers to a specific function by the ADAS system that enables the vehicle to sense and calculate model to determine if the vehicle is moving in the lane properly. The calculation is ongoing and the system generates warning to driver if the LDW module determines that the vehicle is currently outside the intended lane, or is about to cross the lane line, departing the current lane. Figure 8 demonstrates a conceptual system architecture that enables LDW function.

In particular, the input taken from sensors include frames captured from front cameras in real-time, and speed and steering position of the vehicle as captured internally through CAN bus. The incoming frames are consumed by a sensor pre-processor, which uses the frames to recognize objects, such as lane lines, and generates graphical model of driving condition with respect to LDW. The graphics model(s) are then transmitted to the main processor that fuses sensing information from both the graphics models and current conditions of the vehicle, and generates LDW output as the outcome of calculation. The output can be in the form of video/audio to the corresponding output interfaces.

A.2. Adversarial Model

The expected adversary to LDW usage case are the attackers who can launch simple HW attacks with the capability of "university challenge." This means:

- Expected attacker capability: Simple HW attacker
  - Has reverse engineered all firmware, SW
  - Can modify and replace all firmware, SW
  - Can replace/substitute any ADAS components
  - Can remotely install privileged and unprivileged malware onto any micro-processor that communicates through external interfaces
  - Can read/write/jam/forge the radio channel
  - Can add/remove functionality
  - Can boot/operate removed parts in alternate environments
- Expected attacker capability: University Challenge
  - Will invest up to 6 months engineering effort and $50K part/equipment/computation to develop tools to attack many vehicles

A.3. Threat Analysis

A.3.1. Assets and Interfaces

To analyze all threats, the ADAS system is decomposed to assets. Threats on each asset are the threats to the entire system. Assets can be a piece of software, hardware, or data structure. We treat assets as atomic modules in the system. The threats on assets are launched through interfaces, each assets exposed, including input interfaces and output interfaces.

Figure 8 summarizes the relationships between assets. Table I enumerates all assets and the corresponding interfaces.

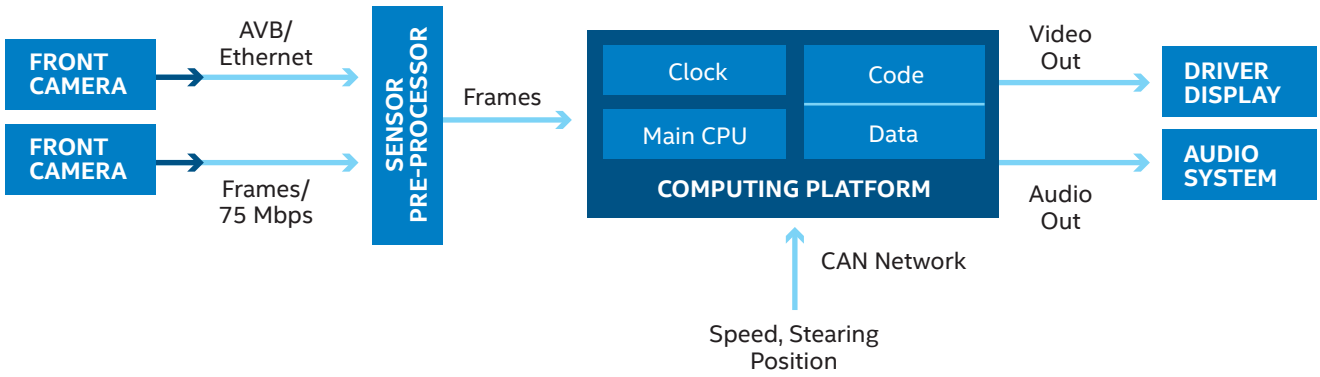

Figure 8. Lane Departure Warning System Block Diagram

ASSET ID	ASSET NAME	INPUT INTERFACE ID	INPUT INTERFACE	FROM ASSET	OUTPUT INTERFACE ID	OUTPUT INTERFACES	TO ASSET
A1	Camera	C1.1	Lens to front view	External	O1.1	AVB/Ethernet interface	A2
A2	Sensor pre-processor	C2.1	Ethernet port	A1	O2.1	Internal bus	A3
A3	Main processor	C3.1	Internal bus	A2	O3.1	Video port	A4
		C3.2	CAN port	A6	O3.2	Audio port	A5
A4	Audio system	C4.1	Audio input port	A3	O4.1	Speaker	External
A5	Driver graphics output	C5.1	HDMI input port	A3	O5.1	Display	External
A6	ECU status reporter	C6.1	ECU specific ports	Internal	O6.1	CAN bus	A3
A7	Frames	C7.1	Ethernet channel	A1	O7.1	Ethernet channel	A2
A8	Graphics models	C8.1	Internal bus	A2	O8.1	Internal bus	A3
A9	Audio warning	C9.1	Audio channel	A3	O9.1	Audio channel	A4
A10	Graphical warning	C10.1	HDMI channel	A3	O10.1	HDMI channel	A5
A11	Velocity info	C11.1	CAN channel	A6	O11.1	CAN channel	A3
A12	Steering wheel info	C12.1	CAN channel	A6	O12.1	CAN channel	A3

Table 1. Assets in LDW Architecture

In this definition, each asset has one or more input interfaces and one or more output interfaces. For the assets that expose some or all interfaces externally, the interfaces are specially labelled. Otherwise, all input interface should specify the assets that the input is taken from, and all output interfaces should specify the assets that the output is delivered to.

Furthermore, assets A1 – A6 are major modules in the LDW system architecture that work together to enable the LDW function. Assets A7 – A12 are specifically defined for the data structure that moves between major functional assets. We define the assets in such way to make sure the properties of the data structures are captured. And unexpected changes to these data structures that alter the properties are considered attacking actions.

A.3.2. Data Structure Asset Properties

For A7 – A12 data structures, we further define all properties of concern.

A7: Frames

- Content parseable as video frame from camera
- Resolution in proper range
- Contrast
- Brightness
- Generated frequency
- Availability

A8: Graphics model data
- Data structure syntax
- Data structure semantics
- Availability
- Generated frequency

A9: Audio warning
- Data syntax
- Data semantics (content delivers the intended meaning)
- Availability
- Timeliness
- Volume

A10: Graphics warning
- Data syntax
- Data semantics (content delivers the intended meaning)
- Availability
- Timeliness

A11: Speed status information
- Data structure syntax
- Data structure semantics
- Availability
- Timeliness
- Frequency

A12: Steering wheel information
- Data structure syntax
- Data structure semantics
- Availability
- Generated frequency

A.3.3. Threat Analysis

To analyze the threats posed by the expected adversaries, we start from the expected behavior of the system. In general, any attack actions that cause the system to act outside its specification, are categorized threats to the system.

The system specification of LDW, from threat analysis perspective, is the following:

- The LDW system should produce accurate warning in time, based on current driving condition.

Therefore, the attacker's goal of causing the system to act outside its specification can be illustrated in the following figure.

As illustrated in Figure 9, the main goal of attacker is "Cause incorrect warning w.r.t. current condition." It can be further broken to three cases:

- False warning: the system produces warning to user, when the vehicle is NOT departing the lane. Or the system produces warning to driver with unexpected properties, which are considered as "false" as well.
- "Removed" warning: the system fails to produce/deliver warning to user, when the vehicle IS departing the lane.
- Delayed warning: the system produces/delivers warning not useful at present time, but may reflect past condition.
  - This threat can be considered as the combination of "Removed" Warning and then False Warning.

The consequences of attacking LDW may include the following:

- False warning:
  - Incorrect warning cause driver panic and potential collision that cause lost of life and damages of properties
  - In-accurate warnings (maybe too many of them) cause decrease user's trust on LDW system, and the user could eventually ignore the LDW system totally

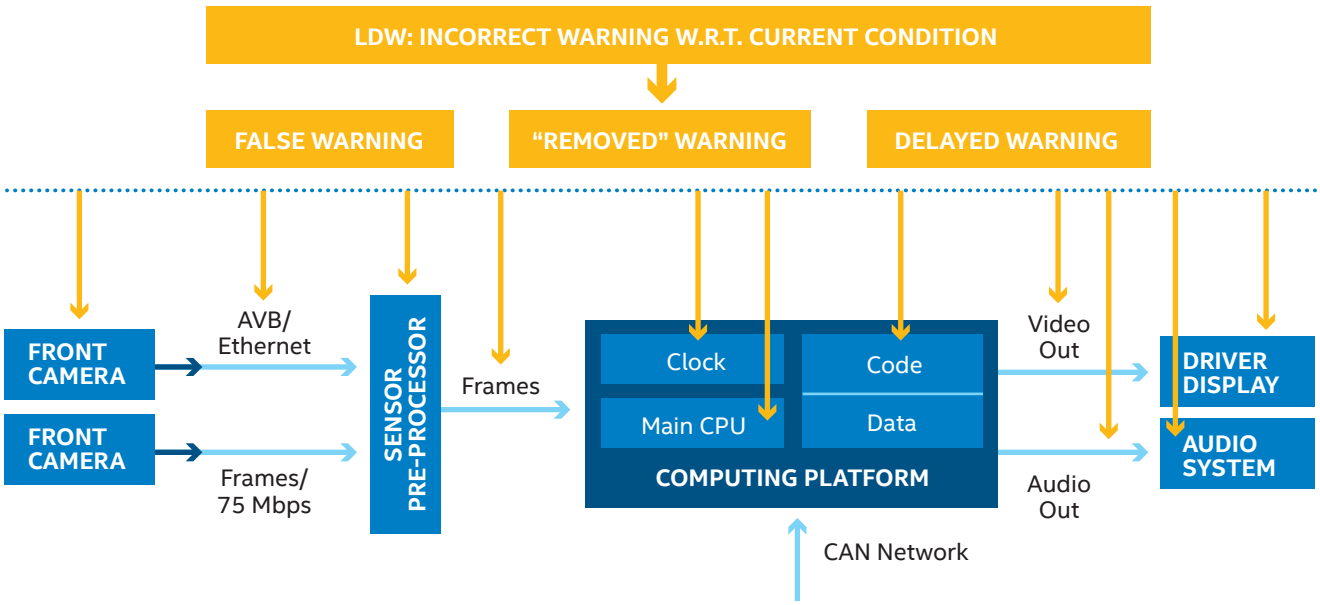

Figure 9. Threat Analysis on LDW Architecture

- "Removed" warning:
  - Lack of warning to reflect current condition is equivalent to the vehicle WITHOUT LDW system.
  - If driver relies on LDW system to react to condition, the driver's reaction to dangerous condition could be delayed
  - Potential collision and/or other fatal consequences
- Delayed warning:
  - Consequences are the combination from False Warning and "Removed" Warning

False Warning Threat Analysis

Table 2 summarizes threat analysis results for False Warning Threat. The threats are categorized into threats on "input data," "internal process," and "output data." The goal is to identify all external/internal interfaces of LDW system assets that attacker can manipulate to conduct actions and cause the system to produce False Warnings.

Summary

- Manipulation of input data properties have direct impact of LDW function correctness
  - Accurately represent current condition (incoming frames and vehicle status)
  - Both content and properties are important
- Internal processes are vulnerable because of
  - Malware threats on sensor pre-processor and main LDW algorithm for fusion
  - Internal communication on Ethernet channel, CAN bus, internal bus, as well as output channels

MAJOR THREATS	SUB THREATS	TARGET ASSETS	TARGET INT	ACTIONS
Input Data	Camera take input	A1	C1.1	• Fake env. condition for camera lens
	Camera process input	A1	N/A	• Insert forged frames directly in camera processing and memory • Infect camera firmware to produce forged frames that indicate lane departure
	Camera delivers frames	A1, A7	C1.1, O1.1, O7.1	• Frames are modified with property change: e.g., blurry frames to confuse modeling software. • Insert forged frameworks to indicate lane departure • Remove frames that indicate “normal” condition
	Vehicle operation status report	A6, A11, A12	C6.1, C11.1, C12.1, O11.1, O12.1	• Block delivery of speed and steering wheel information • Rush/delay delivery of status information • Forge speed information: pretend moving fast to confuse modeling algorithm • Forge steering wheel information: report turning when vehicle moves straight, or report straight when vehicle is turning
Internal Process	Process frames to generate models	A2	C2.1	• Insert malicious code that processes frames and generate models (malware) • Modify internal memory with forged information – Insert frames indicating lane departure
	Output models to further LDW alg.	A2 A8	O2.1, C8.1, O8.1	• Insert malicious code that process output models • Modify internal buffer that stores models – Modify generated models – Delete models – Insert forged models • Internal bus channel manipulation – Same actions as on output buffer
	LDW alg. to fuse sensing data, and generates warning	A3	C3.1, C3.2	• Insert malicious code that fuses sensing data and generates warning (malware) • Modify internal memory with forged information • Insert models that directly indicate lane departure
	Deliver output warning to output devices	A3 A10	O3.1, C10.1 O10.1	• Insert malicious code that process output warnings • Modify internal buffer that stores output warnings – Modify generated warnings with undesirable properties, e.g., turn up warning volume – Re-order buffered warnings – Insert forged warnings in buffer • HDMI channel manipulation – Similar actions as on output buffer • Audio channel manipulation – Similar actions as on output buffer
Output Data	Output device process received data	A4 A5	C4.1, C5.1	• Manipulate input buffer to store forged and incorrect warning data or modify the stored warning data • Malware in warning data processing code to introduce forged final signal for warning output
	Output device delivers warning	A4 A5	O4.1, O5.1	• Malfunction of A4 to deliver unnecessary loud warning • Output HW delivers forged warning directly from attacker

Table 2. Threat Analysis—False Warning Case of LWD

"Removed" Warning Threat Analysis

Table 3 summarizes threat analysis results for "Removed" Warning Threat. The threats are categorized into threats on "input data," "internal process," and "output data." The goal is to identify all external/internal interfaces of LDW system assets that attacker can manipulate to conduct actions and cause the system to not produce warning when it should given the current condition.

MAJOR THREATS	SUB THREATS	TARGET ASSETS	TARGET INT	ACTIONS
Input Data	Camera take input	A1	C1.1	• Fake env. condition for camera lens – E.g., putting up a video to feed in camera lens
	Camera process input	A1	N/A	• Insert forged frames directly in camera processing and memory • Infect camera firmware to produce forged frames that indicate normal driving condition • Delete incoming frames • Delay incoming frames • Reorder incoming frames • Replay normal frames
	Camera delivers frames	A1, A7	C1.1, O1.1, O7.1	• Frames are modified with property change: e.g., blurry frames to confuse modeling software. • Insert forged frames to indicate "normal condition" • Remove frames that indicate lane departure condition
	Vehicle operation status report	A6, A11, A12	C6.1, C11.1, C12.1, O11.1, O12.1	• Block delivery of speed and steering wheel information • Rush/delay delivery of status information • Forge speed information: pretend moving fast to confuse modeling algorithm • Forge steering wheel information: report turning when vehicle moves straight, or report straight when vehicle is turning

Table 3. "Removed" Warning Case of LWD (continued to next page)

Notes: Attacker actions for removed warnings are very similar to the actions for "False Warning" case. All these actions focus on interfaces that could:

- Manipulate input or input processing
- Manipulate output or output delivery
- Manipulate internal processing and internal data communication

As discussed above, the "Delayed Warning" attack can be achieved with combination of actions for "removed" warning and false warning. Therefore, threat analysis for this third case can be considered as union of actions as illustrated in table 1 and 2.

A.3.4. Data Privacy Discussion

So far, we focus our analysis mostly on examine vulnerabilities that cause the ADAS to fail its LDW functionality. Given that the system is expected to constantly collect and process input data from video cameras and internal CAN bus, there might be a concern of privacy of such data. For instance, such data can be used to track location of the vehicle, and reveal behavior history of the driver(s).

In our analysis, we believe, there is no privacy concern if the data is collected only for the purpose of supporting LDW function. The rationale is the following:

- The input data, internal data, and output data are all for temporary consumption. The data is not stored for long term.
- Front view condition is publically available for any one of interest. Instead of attack LDW system directly, attacker could simply install his own camera to capture the same or similar views.

MAJOR THREATS	SUB THREATS	TARGET ASSETS	TARGET INT	ACTIONS
Internal Process	Process frames to generate models	A2	C2.1	• Insert malicious code that processes frames and generate models (malware) • Modify internal memory with forged information – Insert frames indicating “normal conditions”
	Output models to further LDW alg.	A2 A8	O2.1, C8.1, O8.1	• Insert malicious code that process output models • Modify internal buffer that stores models – Modify generated models – Delete models – Insert forged models • Internal bus channel manipulation – Same actions as on output buffer
	LDW alg. to fuse sensing data, and generates warning	A3	C3.1, C3.2	• Insert malicious code that fuses sensing data and generates warning (malware) • Modify internal memory with forged information • Insert models that directly indicate lane departure
	Deliver output warning to output devices	A3 A10	O3.1, C10.1 O10.1	• Insert malicious code that process output warnings • Modify internal buffer that stores output warnings – Modify generated warnings with undesirable properties, e.g., turn down warning volume to zero – Delete warnings directly • HDMI channel manipulation – Similar actions as on output buffer • Audio channel manipulation – Similar actions as on output buffer
Output Data	Output device process received data	A4 A5	C4.1, C5.1	• Manipulate input buffer to remove the warning data • Malware in warning data processing code to delete output
	Output device delivers warning	A4 A5	O4.1, O5.1	• Malfunction of A4 to deliver “muted” warning • Output HW of A5 shut down by attacker

Table 3. “Removed” Warning Case of LWD

Given this reasoning, we think data privacy should not be considered as a threat for LDW system.

However, if in the vehicle ADAS, there is a function module that collects sensing and vehicle status information and stores it in a non-volatile storage for other usages, such as telematics, auditing, etc., then there is the privacy concern that leakage of such information can be used to reveal user’s private driving activities.

Besides data privacy, there may be a need to keeping the details of internal LDW algorithm secret, for vendor’s IP protection purpose. If there is such need, and the algorithm is designed and implemented in such a way that internal data models as input for LDW fusion can potentially reveal algorithm details, there should be mechanisms to keep internal data private, even if such data is only for temporary consumption. Alternatively, the vendor could choose to design or re-design the LDW algorithm, so that internal model data structure won’t provide information that helps to recover algorithm details.

Appendix B. Case Study #2: Adaptive Cruise Control (ACC)

B.1. Adaptive Cruise Control Usage Case

The adaptive cruise control (ACC) use case refers to another specific function by the ADAS system that enables the vehicle to autonomously manage its moving speed, based on the following four sources of information:

- Current vehicle condition
- Driving condition and relative speed of leading vehicle
- Road conditions
- Driver's preference

ACC is the advanced version of current cruise control function. Currently, the cruise control accepts a speed set value from driver and manages the vehicle control system to maintain the target speed. It may takes additional sensing information for speed management from road, e.g., up/down hill and turning. However, traditional cruise control does not allow the vehicle to adjust vehicle speed dynamically according to the relative distance to the leading vehicle.

Primary intelligence in ACC is the capability of recognize leading vehicle(s), estimate relative distance, and even predict the leading vehicle's next moving model.

Compared with LDW, ACC function provides additional opportunity to analyze a more complete example of ADAS function that involves input, internal process, and real-time control command and actuation. Based on ISO 26262 requirements, LDW is generally considered as ASIL B function and ACC is generally considered as ASIL C/D function. In other words, ACC has more stringent safety requirements. For the same ADAS system architecture, implementing ACC means need to reduce exposure to failure, reduce severity of system failures, and increase controllability when failure happens.

ACC function, by nature has higher severity in the consequences of failure, because it involves direct vehicle control, as opposed to only the warning through UI to driver. The controllability is outside the consideration of this analysis. Therefore, our focus should be on exposure to failures. Attack actions to ACC function could potentially increase system exposure to failure significantly. We'll visit this issue in risk analysis.

Figure 10 illustrates a conceptual system architecture that enables ACC function.

The similar ADAS architecture is used to support ACC function. For ACC, the system takes different input data set, runs the ACC modeling algorithm, and output to take actions that manage vehicle speed. Here's the summary:

Input:

- Speed, acceleration status (gas, brake) from CAN bus
- Graphics: front cameras, lidar, radar
- Road condition: surface condition, uphill/downhill, curves, weather conditions (from local sensors as well as through network communication)

ACC Algorithm:

- Leading vehicle(s) object detection
- Distance estimation
- Leading vehicle driving condition assessment and prediction
- Hazard detection: sudden brakes, pedestrians, traffic sign/light, speed limit, road construction, etc.
- Speed management prediction

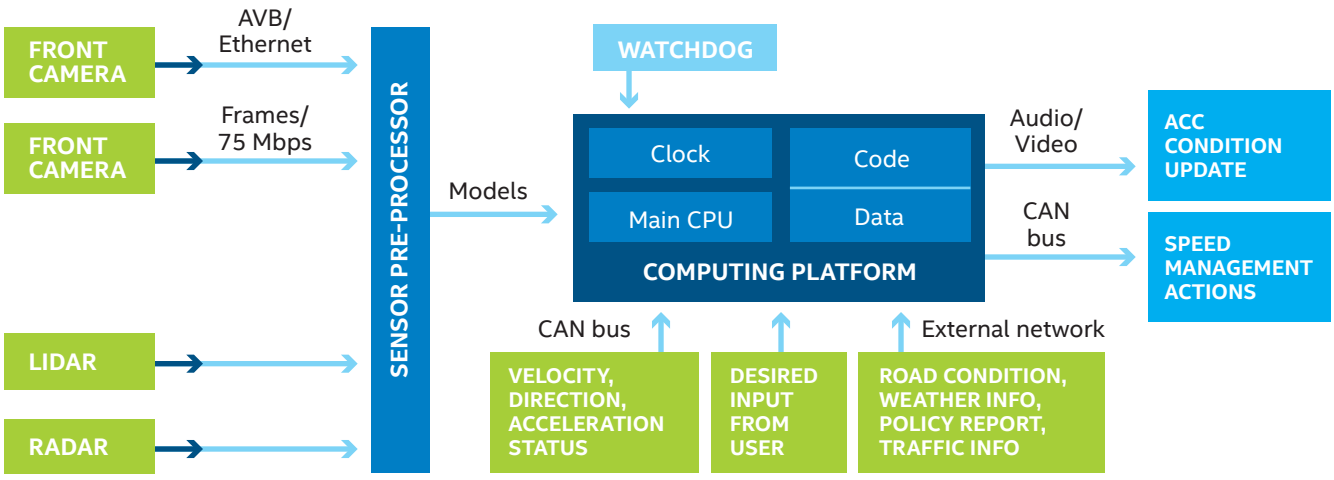

Figure 10. Adaptive Cruise Control System Example Architecture

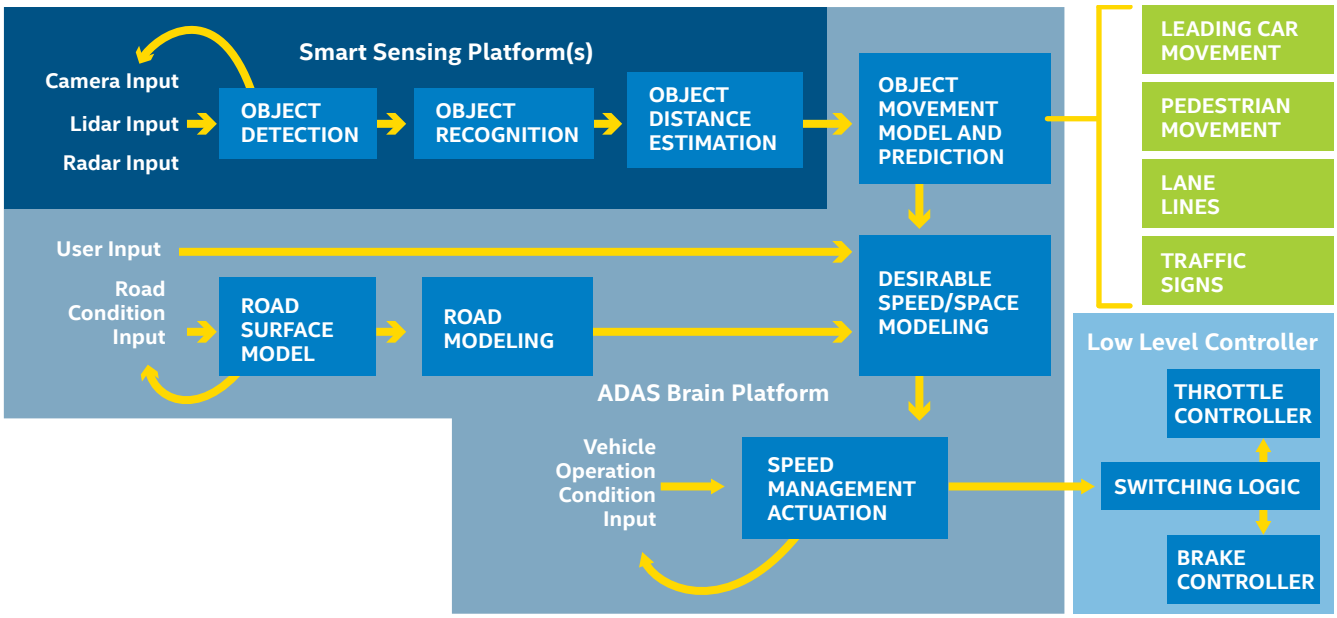

Figure 11. Functional Flow Diagram for ACC

Output:

- Control signals to control speed and adjustment
- Information output through HMI

Figure 11 gives an example flow diagram with respective to more detailed processing in ACC function. Focusing on the fusion algorithms for ACC, there may be a nature separation of processing among the main platform as ADAS brain, and the sensing data pre-processor.

As shown in Figure 11, the smart sensing platform could be responsible for taking input from sensors, and constructing models for object recognition, and extra properties and conditions of the detected objects. In this process, there may be multiple sensing platforms, each handling a set of sensors that provide a specific type of sensing information. In ACC case, sensing processing for video frames from camera, for lidar input, and radar input could be separated. There is, however, a process used to fuse all three types of sensing data to accomplish accurate object detection, recognition, and distance estimation.

The models of recognized objects could then be passed to ADAS brain platform. Here, there are multiple ADAS use cases supported, besides ACC. More powerful fusion process on ADAS brain takes input from smart sensing platform, user's input, road condition input, and vehicle operation status from CAN bus, and conduct processes to:

- Predict object movement
- Model road surface and condition
- Calculate desirable speed for vehicle and/or desirable distance to be maintained from the leading vehicle
- Eventually generate speed management command

The speed management commands are then passed to low level controllers. Here corresponding throttle controller or brake controller processes the input command and take actions to adjust vehicle speed via throttle or brake.

Threat analysis, all these ACC fusion and control processes are under consideration.

B.2. Adversarial Model

The expected adversaries to ACC usage case are the same as the model used for LDW threat analysis.

Here these are the attackers who can launch simple HW attacks with the capability of "university challenge." This means:

- Expected attacker capability: Simple HW attacker
  - Has reverse engineered all firmware, SW
  - Can modify and replace all firmware, SW
  - Can replace/substitute any ADAS components
  - Can remotely install privileged and unprivileged malware onto any micro-processor that communicates through external interfaces
  - Can read/write/jam/forge the radio channel
  - Can add/remove functionality
  - Can boot/operate removed parts in alternate environments
- Expected attacker capability: University Challenge
  - Will invest up to 6 months engineering effort and $50K part/equipment/computation to develop tools to attack many vehicles

B.3. Threat Analysis

B.3.1. Assets and Interfaces

Following the same analysis philosophy, we define ADAS assets for ACC function. The architecture is slightly more complex than the LDW architecture, due to diverse types of input and additional actuation control command output.

Table 4 numerates assets and interfaces for ACC architecture.

B.3.2. Data Structure Asset Properties

For A12 – A21 data structure assets, we further define all properties of concern.

A12: Frames

- Content parseable as video frame from camera
- Resolution in proper range
- Contrast
- Brightness
- Generated frequency
- Availability

A13: Graphics model data

- Data structure syntax
- Data structure semantics
- Availability
- Generated frequency

A14: Audio warning

- Data syntax
- Data semantics (content delivers the intended meaning)
- Availability
- Timeliness
- Volume

A15: Graphics warning

- Data syntax
- Data semantics (content delivers the intended meaning)
- Availability
- Timeliness

A16: Speed status information

- Data structure syntax
- Data structure semantics
- Availability
- Timeliness
- Frequency

A17: Lidar input

- Content parseable as Lidar sensing data
- Resolution in proper range
- Generated frequency
- Availability

A18: Radar input

- Content parseable as Radar sensing data
- Resolution in proper range
- Generated frequency
- Availability

A19: Speed change command

- Content parseable as speed change command
- Desirable speed value within valid range
- Desirable speed value reflects current driving condition
- Desirable acceleration/deceleration time window is a valid range and reflects current driving condition
- Availability
- Timeliness

Assets in Adaptive Cruise Control (ACC) System Architecture

ASSET ID	ASSET NAME	INPUT INTERFACE ID	INPUT INTERFACE	FROM ASSET	OUTPUT INTERFACE ID	OUTPUT INTERFACES	TO ASSET
A1	Camera	C1.1	Lens to front view	External	O1.1	AVB/Ethernet interface	A2
A2	Lidar sensor	C2.1	Lidar sensing port	External	O2.1	Internal communication link	A4
A3	Radar sensor	C3.1	Radar sensing port	External	O3.1	Internal communication link	A4
A4	Sensor pre-processor	C4.1	Ethernet port	A1	O2.1	Internal bus	A5
		C4.2	Lidar link port	A2			
		C4.3	Radar link port	A3			
A5	Main processor	C5.1	Internal bus	A4	O5.1	Video port	A6
		C5.2	CAN port	A9	O5.2	Audio port	A7
		C5.3	Internal bus	A10	O5.3	CAN bus	A7
		C5.4	Network driver	A11			
A6	Audio system	C6.1	Audio input port	A5	O4.1	Speaker	External
A7	Driver graphics output	C7.1	HDMI input port	A5	O5.1	Display	External
A8	Low level controller	C8.1	Throttle/brake command input port	A5	O8.1, O8.2	Internal channel (could be CAN bus) to send command to Throttle Controller or Brake Controller	Internal
A9	Vehicle status reporter	C9.1	ECU specific	Internal	O9.1	CAN bus	A5
A10	On board road condition sensors	C10.1	Sensor specific sensing input port	External	O10.1	CAN bus or other internal channel	A5
A11	Weather condition reporter	C11.1	Internet driver	External	O11.1	Internal channel	A5
A12	Video Frames	C12.1	Ethernet channel	A1	O12.1	Ethernet channel	A4
A13	Graphics models	C13.1	Internal bus	A4	O13.1	Internal bus	A5
A14	Audio warning	C14.1	Audio channel	A5	O14.1	Audio channel	A6
A15	Graphical warning	C15.1	HDMI channel	A5	O15.1	HDMI channel	A7
A16	Velocity info	C16.1	CAN channel	A9	O16.1	CAN channel	A5
A17	Lidar input	C17.1	Lidar sensing port	External	O17.1	Internal link	A4
A18	Radar input	C18.1	Radar sensing port	External	O18.1	Internal link	A4
A19	Speed change	C19.1	CAN bus	A5	O19.1	CAN bus	A8
A20	Road condition sensing data	C20.1	CAN bus	A10	O20.1	Internal channel	A5
A21	Weather information (rain or not, and how hard)	C21.1	Network driver/ port	A11	O21.1	Internal channel	A5

Table 4. Assets in ACC Architecture

A20: Road condition sensing data

- Content parseable as road condition sensing data
- Availability
- Data frequency

A21: Weather information

- Content parseable as weather information
- Generated and provided by an authorized source
- Content authenticated from the authorized source
- Timeliness
- Availability

B.3.3. Threat Analysis

To analyze the threats posed by the expected adversaries, again, we examine the expected behavior of the system. In general, any attack actions that cause the system to act outside its specification, are categorized as the threats to the system.

The system specification of ACC, from threat analysis perspective, is the following:

- Manage vehicle speed adaptive to real-time driving conditions

This means that the success of ACC function can be measured by the following criteria:

- Achieve cruise control function with reasonable conditions
- Minimize danger of collision via real-time speed management
- Minimize traffic congestion factors
- Maximize user comfort

These attacking consequences are illustrated Figure 12.

For the system design, the first two criteria are considered as primary, given the ACC function has direct safety goals. Therefore, the attacker's goal of causing the system to act outside its specification is primarily on: causing the failure to manage proper speed to the point that the attack can compromise the system design goal according to the success criteria.

We are going to focus primarily on the safety critical consequences. The rest of this section presents in more details on the threat analysis that cause ACC safety critical consequences.

Collision Under ACC Threat Analysis

Similar to LDW, the threat analysis is conducted on assets in the system, for threats from "input data," "internal process," and "output data." The goal is to identify all external/internal interfaces of ACC system assets that attacker can manipulate to conduction actions and cause the system to cause collision. A similar table, as LWD threat analysis, can be created for capturing all threats.

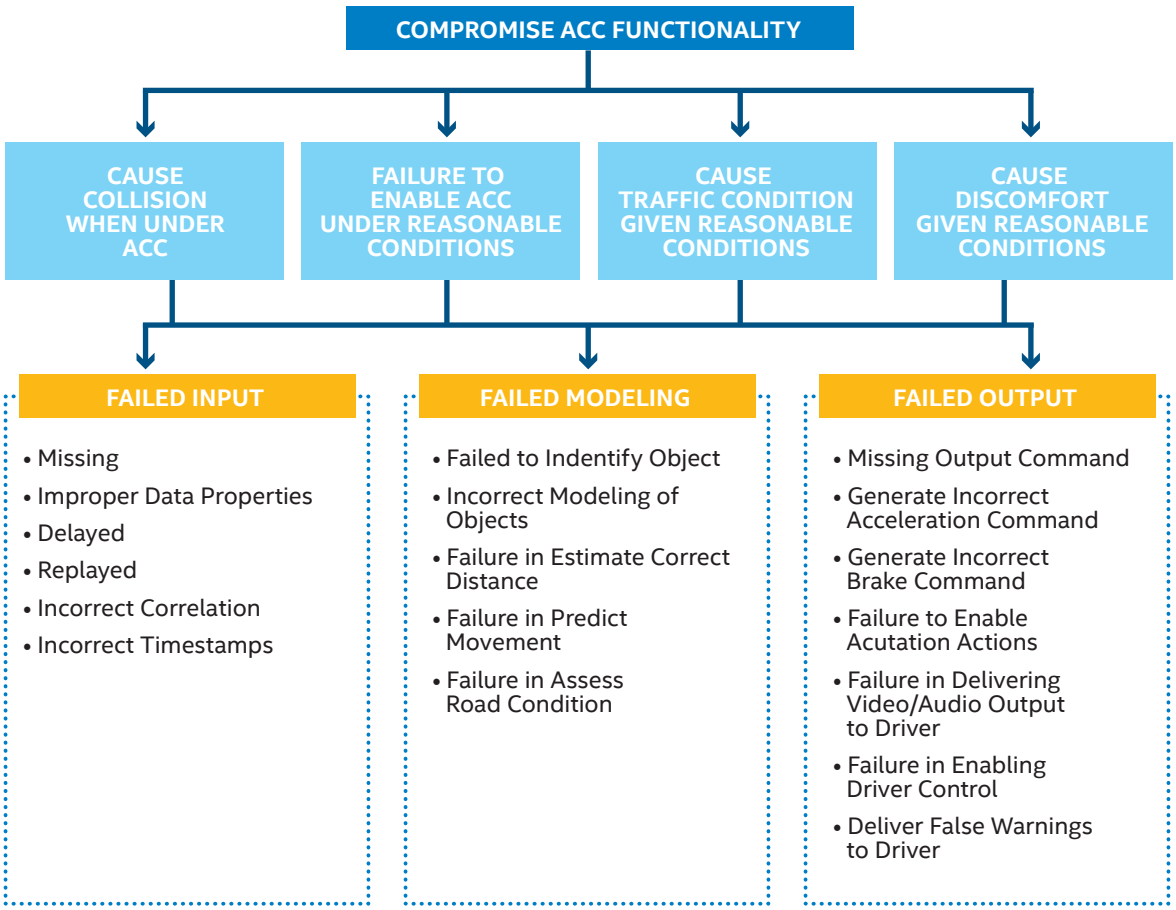

Figure 12. High Level Threats in ACC System

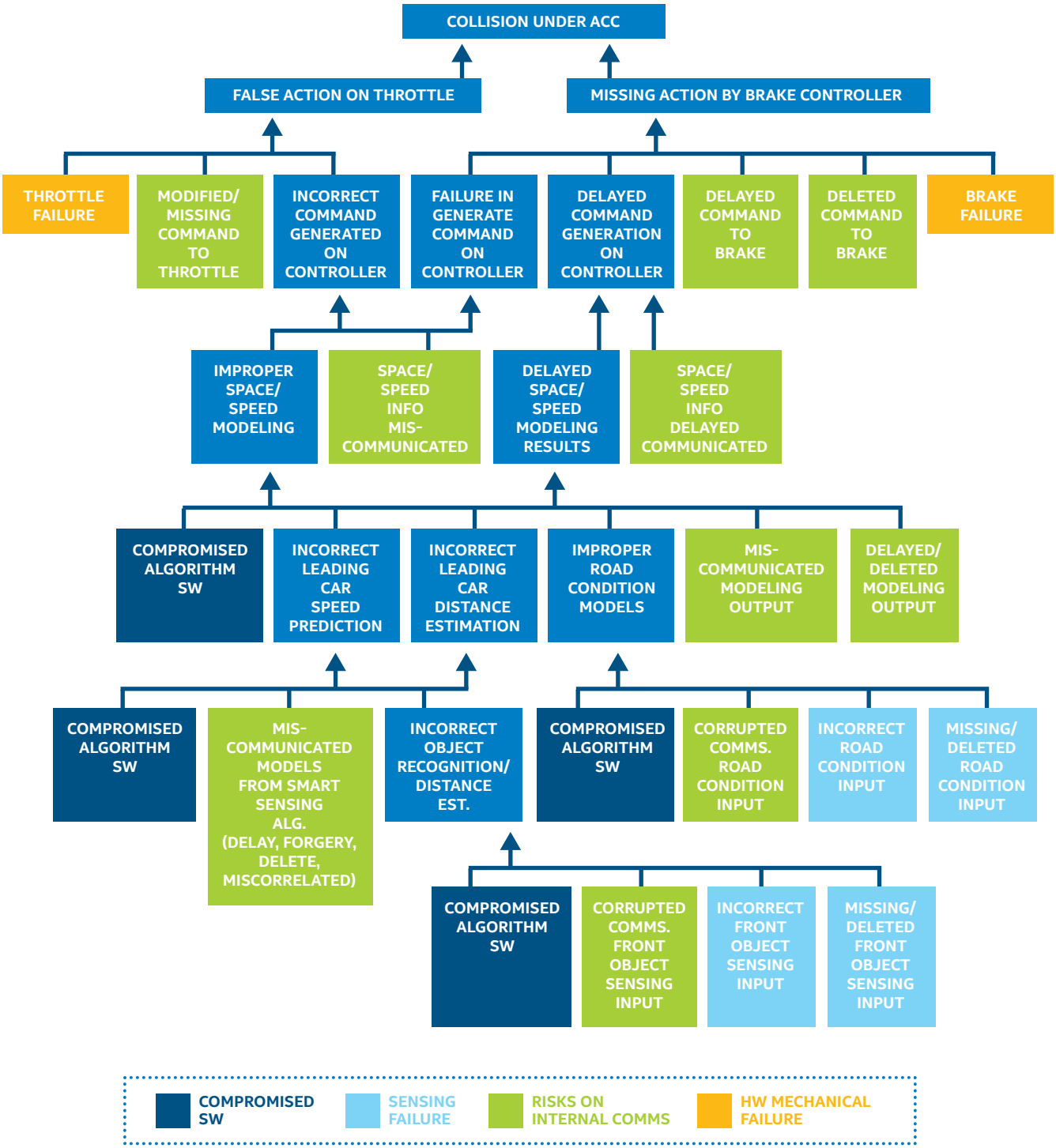

Figure 13. Detailed Threat Analysis on "Collision under ACC" case

Here, we use Figure 13 to summarize our findings. In addition to the enumeration by a detailed threat analysis table, an organized tree of threats is created describe relationship of each threat to how it eventually achieve the attack goals.

In Figure 13, the individual threats are the leaves in the tree. They are color-coded by four categories: compromised SW, sensing failure, risks on internal communications, and HW mechanical failure. The figure illustrates the process how an attack could take action on an entry point, manipulating the system from that leaf upwards to parent nodes, which representing the consequences caused by this action. Eventually, all these internal unexpected changes in the system lead to the final result of compromising ACC function and cause collision.

Here's how the consequences are defined.

L0: Collision under ACC System, caused by

- **L1.1: False action on Throttle,** or
  The primary concern is that the vehicle doesn't slow down in time that causes collision with leading vehicle. Reduction of speed within a time window is the primary concern. Failures in generating, or processing, or executing the appropriate command are the primary causes to this consequence.
- **L1.2: Missing action by brake system**
  When braking is required to avoid collision, missing the braking action becomes fatal. This could be caused by failures in generating, processing, or executing braking commands by the system.

L1.1: False action non Throttle, caused by

- ***Throttle mechanical failure***
  Mechanical problem is out of scope for ACC threat analysis
- ***Modified or missing command to throttle (on A19)***
  False information causes throttle to fail to reduce the speed given required time window. Modification on A19, or the internal channel could include:
  - Incorrect speed target (not enough reduction, or even increase speed)
  - Longer time window specified to reduce speed
  - Missing speed reduction command
  - Delayed speed reduction command
- **L2.1: Incorrect command generated by A8**
  This internal needs further analysis. Incorrect command could be caused by either the input failures, or internal processing failures.

L1.2: Missing action by brake controller, caused by

- ***Brake mechanical failure***
  Mechanical problem is out of scope for ACC threat analysis
- ***Modified or missing command to brake (on A19)***
  False information causes brake to fail to brake in time. Properties on A19 that could lead to the consequence could be either command been delayed or deleted. Further, if there is a time window specified to take braking action, a malicious change to prolong the time window information will cause the failure of braking properly.
- **L2.2: Failure in generating braking command by A8**
  This internal needs further analysis. Incorrect command could be caused by either the input failures, or internal processing failures.
- **L2.3: Delayed in generating braking command by A8**
  This internal needs further analysis. Incorrect command could be caused by either the input failures, or internal processing failures.

L2.1, L2.2, L2.3: Internal: Incorrect command generated by A8, caused by

- ***Miscommunication of target speed information input to A8 (input interface)***
  - Missing speed change, or time window for action restriction
  - Increase of target speed to an undesirable level
  - Delayed speed change command
- **L3.1: Incorrect space/speed modeling results from A5**
  - This is an internal condition that needs further analysis. The desirable outcome by the attacker is to manipulate A5 or the input interface of A5 so that the output speed change command is manipulated to achieve the properties as describe in the above: increase of target speed to an undesirable level or missing speed reduction when it needs to.
- **L3.2: Delayed space/speed modeling results from A5**
  - Similarly, this is an internal condition that needs further analysis. A5 or input interfaces are manipulated in such a way that the output of speed reduction command is delayed.

L3.1: Internal: Incorrect space/speed modeling results from A5, caused by

- ***Compromised ADAS brain fusion algorithm SW***
  This can be achieved by malware infection. Once the fusion algorithm SW is taken by the attacker, its behavior is completely controlled by the attacker. Therefore, the output of space or speed modeling is by attacker's choice completely.
- ***Manipulation on O5.3, for output properties***
  - Properties: speed value, space value, availability, timeliness
- **L4.1: Internal: incorrect modeling of current condition,** including
  - L4.1.1: Incorrect leading car speed prediction
  - L4.1.2: Incorrect leading car distance estimation
  - L4.1.3: Improper road condition models
  - Combination of above

L3.2: Internal: Delayed space/speed modeling results from A5, caused by

- ***Compromised ADAS brain fusion algorithm SW***
  See L3.1 for further details
- ***Manipulation on O5.3, for output properties***
  - Property: delay output
- **L4.2: Internal: incorrect modeling of current condition, that leads to delayed completion**
  L4.1.1, L4.1.2, L4.1.3 could lead to delayed output as well

L4.1.1: Internal: Incorrect leading car speed prediction
L4.1.2: Incorrect leading car distance estimation

- ***Compromised ADAS brain fusion algorithm SW on A5***
- ***Input failures on C5.1***
  - Models delayed, forged, deleted, or miscorrelated
- ***Input failures on C5.2***
  - CAN bus problem to provide correct information on the vehicle speed status
- **L5.1: Internal: Incorrect object recognition or distance estimation from A4**

L4.1.3: Improper road condition models

- ***Compromised ADAS brain fusion algorithm SW on A5***
- ***Incorrect road condition sensing input***
  - E.g., incorrect sensing that misses "downhill" condition
  - Lack of weather info to indicate slippery road on a rainy day
  - This could be achieve by compromising sensors directly, or by compromising external weather source
- ***Missing road condition information***
  - E.g., missing "downhill information"
- ***Corrupted communication to C5.3 and C5.4***
  - Similar properties of sensing data or weather information are changed
- ***Input failures on C5.2***
  - CAN bus problem to provide correct information on the vehicle speed status
- **L5.1: Internal: Incorrect object recognition or distance estimation from A4**

L5.1: Internal: Incorrect object recognition or distance estimation from A4

- **Compromised SW: smart sensing platform (A4) with infected software**
  Output of speed modeling can be of attacker's choice
- ***Compromised communication***
  Input channels are compromised (C4.1, C4.2, and C4.3). Properties could directly affect the outcome of object recognition, and object distance estimation, could include:
  - Availability of frames, or lidar, or radar signals that illustrates current condition
  - Content of input information
    - » Maliciously modified, or inserted with forged information
  - Frequency of input information
  - Correlation of these three sensing data streams that directly impact the outcome of the modeling
- ***Compromised sensing***
  - ***A1: front camera compromised to produce compromised sensing content***
    - » Availability, legitimacy of content, in time
    - » Faked environment on the lens that feed in as sensing input
    - » Compromised camera FW/SW that produces undesirable output
    - » HW compromise on A1, mechanical errors
    - » Failed calibration
  - ***A2: Lidar sensor compromised to produce compromised sensing content***
    - » Availability, legitimacy of content, in time
    - » Faked environment on the input interface to be fed in
    - » Compromised sensor FW/SW that produces undesirable output
    - » HW compromised on A2, mechanical errors
    - » Failed calibration
  - ***A3: Radar sensor compromised to produce compromised sensing content***
    - » Availability, legitimacy of content, in time
    - » Faked environment on the input interface to be fed in
    - » Compromised sensor FW/SW that produces undesirable output
    - » HW compromised on A3, mechanical errors
    - » Failed calibration

Discussion on Other Three Threats

As described above, besides collision caused under ACC function, there are three other types of threats:

- Failure to enable ACC under reasonable conditions
- Cause Traffic condition given reasonable conditions
- Cause discomfort given reasonable conditions

These three threats are real to users, although not necessarily directly related to driving safety. Here we briefly discuss these threats, especially some of the detailed actions by attacker that are not covered in collision threat analysis.

Failure to Enable ACC

Hypothetically, the user input the desirable speed through the existing input interface to trigger ACC function. The ACC system then could take the input, launch the software module, and wake up sensors. The system could be attacked via threats on

- User's input
- Sensor calibration error
- ACC function testing
- ACC function testing output

Under a typical driving environment, such as

- reasonable leading vehicle distance,
- past speed dynamics is reasonable,
- the road is not too slippery, and
- visibility is good enough for sensing,

The ACC can be successfully enabled, after the ACC system runs a round or multiple rounds of quick test. Hence, the threats posed by attacker that can fail the launch of ACC involve actions to cause the system to produce output that indicates failure of one or more of the above 4 conditions.

Overall, the threats can be summarized in Figure 14.

Attack actions on Internal ACC testing procedure can be very similar to the actions taken to achieve collision. Similar threat analysis for collision is applicable here. The difference lies in that the same vulnerabilities are utilized and information is manipulated differently to convince the system that it is too dangerous, or unreasonable to launch ACC now, even though the current driving condition is normal. Hence, the goals are opposite to some of the attacking goals in the collision case.

Failed to launch ACC could also be achieved by failure in accepting user's input, or errors in user input. Attacker could take action on user's input interface to delete, delay, replay, modify, or insert fake command from user. The attack surface could be on the direct external interface, or the module that process the input command, or the internal communication channel to delivers the command to ACC system.

Failed to launch ACC could also be achieved by failing the sensor calibration. During the process of system launching, the necessary sensors need calibration before they can be used to capture sensing data for ACC function. Failure in enabling sensors properly via calibration can directly fail ACC function and system. The attacking surface could be on the direct attacks in sensor mechanics, or the module that handles calibration, or the internal communication channels that pass information between modules for the purpose of calibration, or the output channels to confirm success of calibration.

Finally, output confirmation could be disrupted by the attackers that could effectively disable the launch of the ACC system.

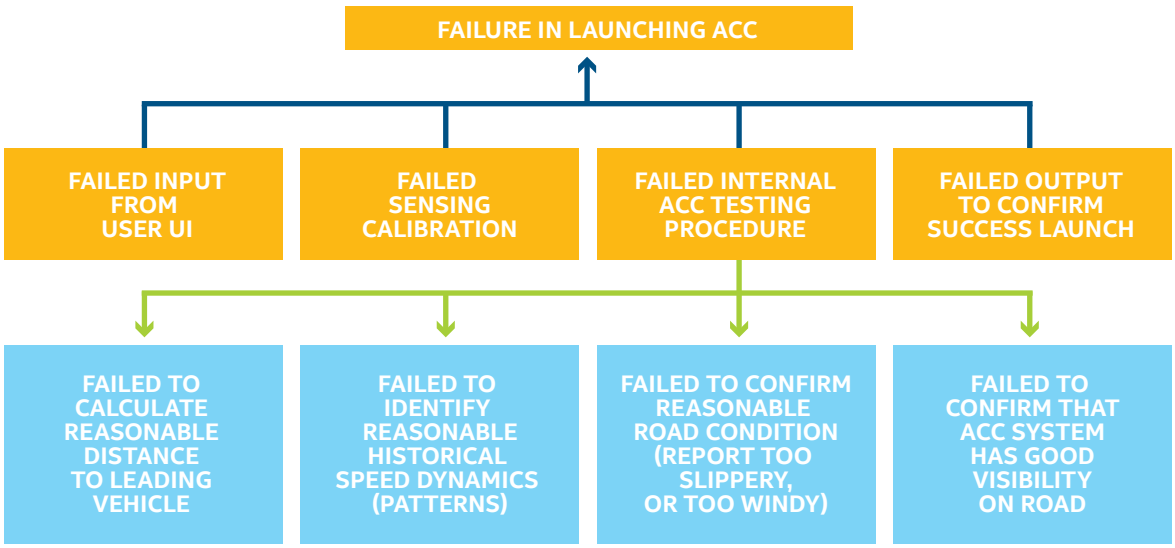

Figure 14. Threat Analysis—"Failure to Enable ACC" case

 0115/MW/HBD/PDF 331817-001US