Towards Safe and Secure Autonomous and Cooperative Vehicle Ecosystems [*]

António Lima, Francisco Rocha, Marcus Völp, and Paulo Esteves-Verissimo
SnT — Interdisciplinary Center for Security, Reliability and Trust
University of Luxembourg
firstname.lastname@uni.lu

ABSTRACT

Semi-autonomous driver assists are already widely deployed and fully autonomous cars are progressively leaving the realm of laboratories. This evolution coexists with a progressive connectivity and cooperation, creating important safety and security challenges, the latter ranging from casual hackers to highly-skilled attackers, requiring a holistic analysis, under the perspective of fully-fledged ecosystems of autonomous and cooperative vehicles. This position paper attempts at contributing to a better understanding of the global threat plane and the specific threat vectors designers should be attentive to. We survey paradigms and mechanisms that may be used to overcome or at least mitigate the potential risks that may arise through the several threat vectors analyzed.

1. INTRODUCTION

The American National Highway Traffic Safety Administration (NHTSA) and the European Commission Directorate General for Mobility and Transport define autonomous vehicles as "those in which at least some aspects of a safety-critical control function (e.g., steering, throttle, or braking) occur without direct driver input" [7]. In this sense, many of our vehicles already have partial *autonomy* thanks to the wealth of x-assists (drive, park, lane, etc.) they incorporate. At the same time, almost all car manufacturers experiment with fully autonomous vehicles and begin gathering kilometers on roads and highways for their safety cases. But not less importantly, *connectivity* has also been increasing, for reasons like infotainment, traffic assistance, remote maintenance or vehicle location. As this trend consolidates, together with a growing infrastructure, many opportunities for *cooperation* between enabled vehicles arise, not only for enhancing autonomous functionality, but also, and in fact mainly, as we explain below, for driving safety.

The main thrust of this position paper is twofold: (i) to propose that we should look holistically at this quickly evolving problematic, as an *ecosystem of autonomous and cooperative vehicles*; and (ii) to perform an analysis of the *global threat plane* that arises from this new reality, and from the specific *threat vectors* designers should be attentive to.

By fitting cars with sophisticated sensors and actuators, and an increasingly powerful network of electronic control units (ECUs), complexity increases, and so does the likelihood of primordial faults, like defects and vulnerabilities. Adding-up connectivity and cooperation further increases system complexity and potentially adds new interaction faults, and overall, all these faults become *exposed* to external threats, such as malicious attacks. This added complexity and openness will amplify the threat plane as seen by vehicles and other ecosystem components (e.g., road-side units). Furthermore, general connectivity and cooperation will become the norm, and we predict that, besides cars, it will be extended to e.g., pedestrians, bicycles, etc., forming a universe of 'sentient' objects, to use the phrasing of an early vision on this subject [17]. In that sense, the autonomous and cooperative vehicles ecosystem may soon emerge as a (virtual) *critical information infrastructure* (CII) with great societal importance. As a consequence, the nature and intensity of threats themselves is also bound to evolve. Further to accidental faults and attacks by casual hackers, we should expect advanced persistent threats and targeted attacks mounted by highly skilled and well-equipped adversaries.

In this paper, we contribute with a global threat plane definition and a fairly complete set of threat vectors, which go well beyond isolated cars, including road-side units, trusted authorities, other enabled cars, and very importantly, the new kinds of interactions that arise in this ecosystem. Past literature has made quite important cases about existing vulnerabilities and their successful exploitation, in specific sectors of this ecosystem, such as Al-Kahtani [9], Mejri et al. [47], Al-Sultan et al. [10], di Pietro et al. [22], and Mokhtar and Azab [49]. Further studies can now draw on those very useful contributions, in the context of our threat model, to assess the likelihood and impact of successful attacks through specific threat vectors, and how these risks can be mitigated by the solutions we propose, drafted under the perspective of attacks being performed by both casual and highly-skilled adversaries.

In Section 2, we make the case for a holistic approach to, respectively, cooperation and autonomy, and safety and security, motivating by examples. Section 3 explores the safety-security gap with insights into recent events. Sec-

[*] This work is supported by University of Luxembourg - SnT and by Fonds National de la Recherche Luxembourg (FNR) through PEARL grant FNR/P14/8149128.

CPS-SPC'16, October 28 2016, Vienna, Austria

ACM ISBN 978-1-4503-4568-2/16/10…$15.00

DOI: http://dx.doi.org/10.1145/2994487.2994489

tion 4 introduces the autonomous and cooperative vehicles ecosystem. In Section 5, we present the threat model used, and the threat vectors it conveys. After we analyze the threat vectors, we sketch current and possibly future solutions in Section 6. Section 7 concludes this article.

2. THE CASE FOR A HOLISTIC APPROACH TO THE SAFETY AND SECURITY OF AUTONOMOUS AND COOPERATIVE VEHICLES

Imagine a driver seeing a woman on the side walk and suddenly hitting the brakes to stop before a child runs after a ball that was rolling onto the street. Did he see the child or the ball? No. But from the gestures of the woman he deduced apparent danger and proactively stopped his car. Likewise, imagine that you are about to slowly enter an intersection with right-of-way (priority-to-the-right) and you establish eye contact with a faster driver coming from your left, and from the mutual gestural language both achieve consensus that you yield to her. Could both scenarios above be possible if protagonists were current first generation autonomous cars, which are individualistic (non-cooperating)? Very likely, no. On the contrary, in the recent accident of a Google car with a bus in a lane change conflict [6], both the car driving program and the bus driver, acting individually, decided to occupy the same spot in the lane at the same time. This further shows that not only may cooperation improve traffic management, but the lack thereof may be an impediment to safety.

Eventually computer systems will understand these implicit forms of communication. But until then, these and similar situations will occur, which for the foreseeable future are hard to tackle in an individualistic way. The situation is certainly bound to get worse when (individualistic) autonomous cars of different makes start meeting on the road. So, we should consider the next best technical solution, which is to have autonomous vehicles and other ecosystem entities cooperating through explicit communication: preceding cars, which might have been able to detect the kid playing, or better, a smart band around the child's wrist, detecting his trajectory and beaconing to the car; or performing consensus among several cars negotiating an intersection, or changing lane. Naturally, there is a long road ahead, for example, standardization of exchanged data and the security of the communication infrastructure are major challenges to realize this ecosystem vision.

We predict that cooperative driving will soon be intensively in the agenda of autonomous driving stakeholders, since it is a powerful way to improve safety, and brings an additional set of functional benefits. However, cooperation in this context means opening-up cars, road-side units and other components of the ecosystem (wireless communication will be a workhorse of this reality), increasing the threat surface of these systems, not only to casual hackers, but also to highly-skilled and well-equipped adversaries, capable of perpetrating targeted attacks.

Imagine coordinated attempts to break into autonomous cars and cause serious accidents, or to use them for blocking roads for criminal or terrorist actions. Science fiction? No. In the past, we have already seen successful attacks, and several researchers [38, 43, 55, 19] have already identified this kind of horror scenarios. Miller and Valasek [48] already mounted a successful car attack, compromising the cellular and Wi-Fi communication to shutdown the engine, disable brakes and lock the doors. Law enforcement agencies, for example, have already issued warnings about the increasing possibility of vehicular cyber attacks [3].

While law initiatives and safety regulations may help prevent a good deal of safety violations and casual attacks, a subtle issue may contribute to obscure the perception of reality. Traditional safety-case analyses may rightly assume that the residual probability of faults in a well-designed car control system leads, under an accidental fault threat model, to an infinitesimal and acceptable probability of catastrophic failure. We cannot help recalling the recent (one dead) accidents with Tesla autonomous cars [5, 4], allegedly due to safety failures in the autonomous driving control system, and conjecture that possibly the above-mentioned safety-review goals have not been met.

However, whatever a defect/fault is, whose activation is capable, even on very rare occasions, of causing a catastrophic failure, such as in the sad events above, then, in the presence of intentional and malicious adversaries, the known phrase "if it can happen, it will happen" risks being fulfilled, defying stochastics. This *safety-security* gap may explain why so many successful attacks could have been deployed, in an extremely solid and dependability/safety concerned industry. So, in what concerns autonomous and connected or cooperative vehicles, we argue for a change in the mindset behind threat models and safety cases, when malicious adversaries are part of the equation: security and safety must go hand in hand from the first hour, both when analyzing what may go wrong (safety I) and when ensuring correct operation despite faults and attacks (safety II).

The threat model must equate simultaneously accidental and malicious threats, as we propose in this paper, it being the case that malicious threat vectors will most of the times superimpose themselves to accidental ones. Likewise, the likelihood of safety (catastrophic) failures must be analyzed under the perspective of both accidental and intentional causes. For the latter, if systems are not hardened by construction to proactively achieve resilience despite attackers exploiting safety vulnerabilities, stochastics play a moderate role in the way of protection.

3. SOME INSIGHTS INTO THE SAFETY-SECURITY GAP

The many crashes due to unintended acceleration (UA) of Toyota vehicles, happened lately in the US [18], allegedly killed almost 90 people in the last decade. They yielded an extensive investigation and trial [51] and as such became an interesting case study, providing relevant insights onto our argumentation. Note that these vehicles are in the lower step of the autonomy ladder: the higher up we go, the more important this problem becomes.

Before we start, let us align in abstract the logically possible causes for each of these accidents:

1. Accidental UA - and then, either: (a) directly due to a specific mechanical, electrical or electronic/computing defect; or (b) due to a weakness of fail-safe mechanisms, which, once activated by some generic electronic/computing defect, namely a software bug, would fail to prevent leading the vehicle into UA. Given the high repeatability of electronic/computing defects, it

should be relatively easy to find both deviations of known good behaviour possibly causing 1(a) and the weakness in 1(b), targeted to the fail-safe mechanisms, despite possibly initiated by unknown causes.

2. Provoked UA - and then (we confine ourselves to electronic/computing defects, because of the repeatability), either: (a) through the exploitation, by malicious perpetrators, of a safety-impacting software defect directly leading to UA; or (b) the exploitation of a (set of) non-safety-impacting defect(s), by malicious perpetrators, which would allow them first access to in-car networks and/or ECUs and from there, to disturb the control (and even the fail-safe mechanisms) by a direct attack either on the buses schedules or on the ECU code, ultimately leading to UA. Even given the high repeatability, it might not be easy to find the cause of 2(a). As for 2(b), even without finding concrete attack possibilities, a symptomatic study of the quality of software might allow assessing the likelihood of this scenario.

In the context of the trial investigations, some experts (National Highway Traffic Safety Administration (NHTSA) and NASA) thoroughly reviewed both hardware and software, and concluded that they had found no evidence that mechanics or electronics were to blame for the alleged UA. Defects might still be there, but they would have to be extremely subtle, the kind of 'heisenbugs', or sporadic faults.

In any of the two latter cases anyway, the safety failure scenario leading to UA should be too improbable to be repeatable by accident several dozens of times out of these primordial causes. This would seem to make 1(a) or 1(b) of accidental UA less likely. Nevertheless, another group of experts (Barr) reviewed the source code with static analysis tooling and diagnosed a relevant possibility of generic defects existing in the code, without specifically identifying them. Certainly, this brings back 1(a) or 1(b) as possibilities, due to an ample conjunction of faults.

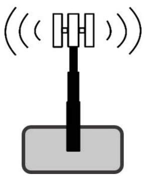
However, let us not forget the provoked UA scenario. The first analyses cited may also seem to make 2(a) less likely. However, the findings of Barr objectively point to a reasonably large degree of vulnerability (in relative terms, with high safety standards in mind). Even without any safety-impacting defect in the car under an accidental safety-case scenario, i.e., in absence of 1(a), 1(b), *and even* 2(a), the vehicle may still present a risk of provoked safety failure, if a sufficient number of generic vulnerabilities exist, especially zero-day vulnerabilities, because they may allow staging targeted attacks to compromise otherwise correct components of the system and, in the end, ultimately lead to UA.

Note that we are making *absolutely no suggestion* about what might have happened, but just making a logical analysis of the odds and possibilities, destined as to prove our point about the importance of closing the *security-safety gap* in vehicle systems.

4. THE AUTONOMOUS AND COOPERATIVE DRIVING ECOSYSTEM

We enumerate the components of the autonomous and cooperative driving ecosystem we envision and briefly describe their functionality. We consider that all active components of the system: (i) are *connectivity-enabled* with the protocols agreed by the system; (ii) have in addition, for those that are *autonomy-enabled*, incremental levels of 'sentience' [17], through higher-level protocols and middleware endowing them with the capability of acting autonomously on information acquired from other sentient components or the environment, directly or indirectly, and eventually cooperate with other sentient components. The remaining abstract components are considered 'environment'. They include for example, legacy, i.e., non-enabled cars, bicycles, pedestrians, etc. The figures depict the different actors we describe below.

s-Car. An enabled, *sentient car* with own sensors, in-vehicle communication networks, protocols and actuators enabling the overall operation of the vehicle. It interacts with other sentient components via external communication channels and protocols and with the environment via its sensors and actuators. Whilst connected and sentient, it drives with a level of autonomy that can range from level zero (no automation) to level four (Full Self-Driving Automation) in the NHTSA autonomous vehicle level definition [7].



Road-Side Unit (RSU). It bridges communication between s-Cars, and with further resources, e.g., the Internet, or remote clouds. It can also act as a SCADA-like system[1], coordinating and acquiring information from sentient vehicles, in order to build near-real-time images of the state of traffic or roads in the area.

Trusted Authorities (TA). The system's *roots-of-trust*, certifying directly, or indirectly, RSUs, s-Cars and other s-Components. Certification is achieved by validating member identities. TAs can be in the hands of public or private organizations, e.g., NHTSA, TÜV or car manufacturers.

s-Components. As the ecosystem evolves, we predict the appearance of other enabled, sentient components, such as bicycles and motorcycles, but also enabled pedestrians. Safety impact can be enormous, if for example, s-Cars and the former can exchange location and motion data.

Environment. Includes everything that is not connectivity-enabled for a start, i.e., which are not so-called active components. In consequence, the physical environment itself of course (roads, fixed obstacles, etc.), but also 'legacy' cars and other non-enabled, bicycles, pedestrians, etc.

In summary, the active components described above, form the autonomous and cooperative driving ecosystem, whose architecture is depicted[2] in Figure 1. S-Cars connect with other s-Cars and s-Components through vehicle-to-vehicle communication ($V2V$) to exchange information sensed from

[1] Supervisory Control and Data Acquisition.

[2] Let us ignore for now the threat vector arrows, whose meaning will be explained in Section 5.

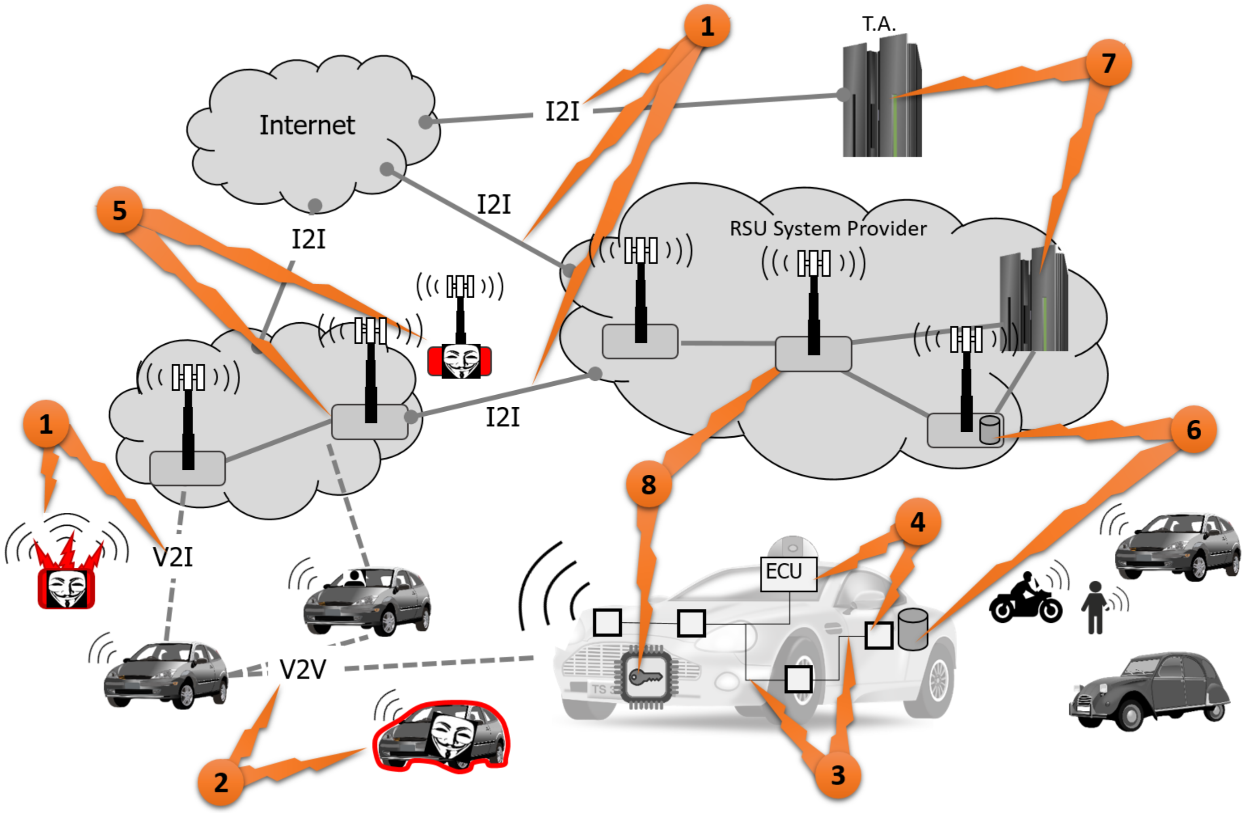


Figure 1. Autonomous and cooperative driving ecosystem architecture (and relevant threat vectors).

the environment and self-learned driving information (e.g., the trajectory of a car having overtaken part of the platoon). Vehicle-to-infrastructure communication (*V2I*) connects mobile s-Cars and s-Components with stationary roadside units such as intelligent traffic lights, the base stations, which are also used for mobile phone communication, and other systems within wireless reach.

RSUs are connected through private networks and to the Internet, therefore they may bridge between physically apart ad-hoc vehicle networks (VANETs), and between s-Cars and central servers. We call this connection between infrastructures infrastructure-to-infrastructure communication (*I2I*). RSUs may improve situational awareness (bad condition, debris, or obstacles on the road are typical examples of SCADA-like state that can be passed from s-Cars detecting them, to all others). The environment is not connected and hence not shown in the figure: information about the environment must be sensed or received from other connected components having sensed the environment. We now turn to the security analysis of this ecosystem.

5. ANALYSIS OF THREAT VECTORS

In this section we describe general security and safety expectations, together with our threat model and threat vectors to which the ecosystem in Figure 1 is exposed. To protect passengers and other traffic participants, autonomous vehicles must preserve the following six properties in the presence of both accidental faults and malicious attacks: (i) correctness of the vehicle's sensing, processing and actuating functions; (ii) correct and timely transmission of information over several kinds of networks; (iii) integrity of the control-state related data, generated and stored in the vehicle's ECUs or in the RSUs' SCADA systems; (iv) avoidance of single points of failure and generic tolerance of faults and intrusions; (v) resilience against partial failures and continued compromises, as well as prevention of their escalation through the ecosystem; and (vi) avoidance of over-sensitive fail-safe shut-down mechanisms.

As with any other information infrastructure dealing with personally identifiable information (PII), privacy also becomes an increasingly important protection goal in the autonomous and cooperative vehicles ecosystem. For example, regular V2I communication allows RSUs to track the location of individual vehicles and their owners, which raises privacy concerns.

Next we describe the threat model we are using in this paper, taking into account the considerations made earlier: autonomy and connectivity/cooperation; accidental faults, casual hackers, targeted attackers. We derive a list of threat vectors and superimpose them on the architecture of Figure 1.

5.1 Threat model

Technically, s-Cars, s-Components, RSUs and, to a limited degree, central servers, form a distributed real-time system [40]. That is, both the correctness of values and their timeliness have to be preserved. Timing assumptions are formulated at different scales, from hard real-time x-by-wire and engine control loops in ECUs with timings at the scale of a few milliseconds, to near-real-time in the traffic information functions of RSUs, which are able to tolerate delays of a few seconds. We assume correct s-Cars and other s-Components to follow common security and safety principles and agreed protocols.

There are several levels of openness in the networks involved, ranging from: the Internet, which is public; the RSU intranets, which are private, but geographically exposed, exhibiting wireless connections; the V2V networks, which are public and wireless; in-car wireless networks; and the least accessible, in-car cable networks, requiring either physical access, or a first jump through the barrier between the car wireless network gateway, and the remaining in-car networking infrastructure. We assume threats may be deployed by four categories of threat agents:

External (X): e.g., computers on the Internet, compromised RSUs, hostile s-Cars or hostile computers near s-Cars;

Local (L): compromised or hostile computers inside s-Cars, connected media (e.g., rogue USB sticks).

Physical (P): compromised or hostile computers on maintenance sockets, rogue humans replacing or inserting hardware; supply chain subversion.

Environmental (E): devices interfering with the physical environment properties (e.g., jammers disturbing wireless transmissions or similar devices leading to creating false sensor images); fake RSUs.

In this paper, we do a holistic analysis that builds the global threat plane emerging not only from specific components of the system and the vulnerabilities they may expose, but also from new, complex interactions originating in a fully-fledged ecosystem. We also take into account that it can be hit by accidental faults and attacks by casual hackers, as well as by advanced persistent threats and targeted attacks. Drawing from the well-known AVI — attack, vulnerability, intrusion — composite fault model [73], we decouple the existence and accessibility of defects and vulnerabilities, from the faults and attacks that may activate them, and which lead to the view of the global threat plane. Although we define individual threat vectors, we are aware that real adversaries may, and will, conjugate threat vectors and apply multiple agents to mount intrusion campaigns to achieve their objectives.

In the definition of threat vectors, we do not impose bounds on the leverage of threat agents, with the exception of computational bounds such as the infeasibility of brute-force attacks against encryption keys. Attackers may eavesdrop on communication or become active and inject messages whenever this is feasible. We believe such an approach generalizes the analysis. Given the defined threat plane, further studies can easily investigate how likely and severely specific vulnerabilities (e.g., those found in previous studies) can be hit, through the vectors described below and summarized in Table 1.

5.2 Threat Vector #1: Attacks on global communication

External entities and also local ones if compromised, may be used to attack the global infrastructure for V2I and I2I communication. Attacks include: replaying messages (e.g., of an emergency vehicle to gain priority road access); distributed denial of service (DDoS) attacks or name server attacks to temporarily bring down or disconnect part of the infrastructure (e.g., TAs or home nodes used for authenticating cars); and forgery of messages and identity attacks to create a false perception of the environment [34] or to simulate different identities [23, 50]. Some of the above effects can also be achieved through an attack by an environmental agent jamming the wireless V2I signal.

For attacks to I2I infrastructure, moderate adversarial effort suffices to e.g., mount DDoS attacks against servers connected via the Internet. V2I attacks are more demanding, because RSUs are connected through intranets, which require an indirect attack or tampering with the authentication mechanisms (see Vector #7) to mimic an RSU.

V2I and I2I attacks may have a simultaneous effect on a large number of s-Cars and s-Components. Through them, adversaries may cause immediate disruption in a large region (e.g., by blocking all access routes to a crime scene).

5.3 Threat Vector #2: Attacks on local V2V communication

In addition to the infrastructure-based attacks, V2V inherits the attack possibilities from ad-hoc networks. Hence, we have to consider network topology and the ability to shield individual vehicles from the majority [37]. However, we consider this attack vector less severe because of the following two reasons: (i) we expect RSU infrastructure to be present in all critical places where traffic volume dictates high speed or low distance in difficult road situations — in these cases, threat vector #1 applies; and (ii) in those situations where no RSUs are present, traffic is more relaxed to allow vehicles to travel at a velocity and distance where autonomous functions have time to take over. Vector agents can be local or external (e.g. resp., a compromised node in the vehicular network or a rogue s-Car) or even environmental.

5.4 Threat Vector #3: Attacks on in-vehicle communication

In-vehicle communication combines critical and not so critical messages in the same network, separated through gateways and fail-over mechanisms. Communication is based on bus-based systems such as LIN [59] for sensor and actuator to ECU communication, MOST [2] for multimedia, FlexRay [20] and CAN [53] for low-speed comfort and high-speed tasks such as anti-lock breaking, engine control and gears, and time-triggered Ethernet and wireless connections (e.g., to signal tire pressure). Networks are typically wired and attacks require physical access to the car (though unfortunately not always its interior [77, 67]) or compromised local agents. But once inside, we point-out two flavors of this attack vector: attacks on ECUs (e.g., by injecting messages to the car's buses to stop the vehicle or to change the engine's operation mode); and attacks on message timing. For high speed buses such as Controller Area Network or time-triggered Ethernet, late sends and artificial message delays can be catastrophic as it may break the synchronization between CPU and network schedules, which in turn may result in late event processing. Alien local or physical agents can see their infiltration made possible by insufficient trust and control access enforcement.

Vector	Description	Agents
#1	Attacks on global V2I/I2I communication infrastructure	X, L, E
#2	Attacks on local V2V communication infrastructure	X, L, E
#3	Attacks on in-vehicle communication infrastructure	L, P
#4	Attacks on vehicle computing nodes' software	L,P
#5	Attacks on road-side units'software	X, P, E
#6	Attacks on sensors and control-sensitive data	X, L, P, E
#7	Attacks on authentication mechanisms	X, L, P
#8	Physical-level attacks	P

Table 1. Threat vectors on autonomous and cooperative vehicle ecosystems

This threat vector and the following two form the attack surface of classical autonomous but non-cooperative cars. Attacks at this level are a prerequisite in order to control and/or prevent action from local controls and corresponding failsafe mechanisms (e.g., as a response to failures/attacks at the cooperative level (Vector #1 and #2)).

5.5 Threat Vector #4: Attacks on exposed vehicle software

Computing resources in autonomous cars are single- and multi-core systems in ECUs, but in order to support the increasingly complex and demanding autonomous and cooperative driving tasks, we have increasingly graphics-based systems (GPUs), FPGAs, high-end multicore systems, and even standard PC-like platforms for infotainment. Tasks in the latter systems will run on legacy operating systems such as Ubuntu Linux and as such inherit all vulnerabilities from PC systems. Even in the more robust control ECU systems, vulnerabilities may loom, given the increased complexity and frequent in-field update cycles in the desire to sell and add new functionality. Likewise, due to the equitable access of ECUs to in-car buses, compromise of an ECU yields arbitrary access to the bus. Once compromised, injecting fake sensor data to a control loop on a remote ECU is trivial [19, 41], in particular if the identifier is not hardcoded to allow multiple identifiers and hence transmission priorities per source. Some of these systems already provide rudimentary safety-oriented resilience, however, these measures do not suffice against malicious nodes [60]. Local and physical agents are those who normally operate through this threat vector. On the physical side, one should further have in mind supply-chain subversion, i.e., compromise of a part before it is inserted in the car (e.g., during fabrication).

5.6 Threat Vector #5: Attacks on exposed road-side unit software

RSUs serve as access point for V2I communication and as bridge to the Internet and other infrastructures. As such they show many similarities to mobile phone base stations and inherit the threats they are exposed to. External agents (from the perspective of the RSU) may penetrate RSU-internal firewalls to gain access to the V2I network, to compromise computation code (e.g., of the consolidated traffic situation) in order to signal false information to connected vehicles. Furthermore, these threats can be consummated as well through physical agents accessing the (normally unattended) premises. Environmental agents in the area may also set up fake RSUs [54].

5.7 Threat Vector #6: Attacks on sensors and control-sensitive data

This threat vector is concerned with the manipulation of data needed directly or indirectly for control. Depending on the position and leverage of the threat agent, it may not require an active attack on control apparatus or server software, but relatively low privileges on data repositories or producer/transmission devices. It consists in manipulating control-sensitive data in a way that 'correct' algorithms and software produce 'incorrect' results. We foresee in-car data, such as imaging, or RSU SCADA data, such as traffic state, as potential targets of this vector. The example described in [70] is, as the name says, intriguing, since it shows that manipulated images which are, to the naked eye, no doubt, vehicles, can fool the machine learning software modules. This is an excellent illustration of the security vs. safety gap of automobile safety cases, to which we alerted in Section 2: malicious manipulation of autonomous driving image recognition can cause crashes which emulate accidental failures; likewise with RSU near-real-time traffic information, where a skilled manipulation attack may induce operators or control software to take wrong decisions, wrecking havoc in traffic. Control data may also be tampered with by environmental agents interfering with sensors (e.g., LiDAR) detection [56]. In hindsight, although it seems that the recent Tesla car crash [5] was provoked by an accidental failure of image processing software, the defect is there, and even if the probability of activating that defect were much more reduced, so that the crash would "never" happen by accident, a motivated and skilled attacker might probably find a successful attack along this vector.

5.8 Threat Vector #7: Attacks on authentication mechanisms

Trusted Authorities (TA) are key in establishing trust between the different vehicles and road-side units. A centralized trusted service is a valuable principle since it enables techniques to become simple and efficient by basing their trust on the TA. The TA is usually responsible for assigning, managing and revoking permissions of network members. Much like in the public-key infrastructure, Trusted Authorities become single points of failure. The whole system is built on trust provided by the TA. One of the problems is assigning who would be in charge of certification. Governments, car manufacturers and chip producers are some of the entities that could be involved in the process.

Like in Vector #1, control of a Trusted Authority and corresponding secret keys used in authentication protocols enables adversaries to mount large scale physical attacks by installing and attesting rogue infrastructure.

5.9 Threat Vector #8: Physical-level attacks

Hardware-level attacks are a delicate matter and should be considered in any system that depends on a hardware layer correctly implementing its specification. Since the software in autonomous vehicles and RSUs executes on top of hardware, compromising that same hardware exposes the layers that depend and trust on it to execute its operations. RSUs and autonomous vehicles are likely to be physically accessible to adversaries that have time to plan and execute attacks against these entities. Attacks, coming essentially from physical agents, might range from complex design or fabrication time attacks to the installation of a malicious device on a vehicle's on board unit [79].

6. TOWARDS SAFE AND SECURE OPERATION

In this section, we summarize solutions and review related work, in terms of mechanisms and paradigms to mitigate the potential risks that may arise through the threat vectors analyzed in the previous section. Table 2 lists the main vectors covered by each of these solutions, as well as the security and dependability properties secured or improved.

6.1 Trust Management and Access Control

One if not the major hurdle on the way towards dependable autonomous and cooperative driving is the longevity of vehicles and the implied longevity of built in hardware and software, whose upgrade gets hampered by certification requirements. To partially mitigate this problem within cars, manufacturers strive for more dynamic, upgradable systems granting them the possibility of in-field software upgrades and, through replacements or plugable extensions, also of some hardware parts.

Less clear is the trust management of and access control to data held and processed at RSUs and more remote infrastructures. But surely, multiple competing stakeholders have an interest in accessing partially private information in the data sets received. We can imagine co-migrating data vaults, owned by the car's manufacturer or its owner, who collects, represents and grants access to data gathered from the vehicle. Alternatively, RSUs could be under the responsibility of government subcontractors as happens with traffic lights and similar control systems today. Technical solutions for policy and trust enforcement and supply-chain control already partially support these solutions. Any other solution discussed in this section requires trust management and access control policies. That is illustrated in Table 2 with a special symbol reserved for this type of solutions.

Schlatow et al. [64, 63] define a framework for reconfiguring complex systems while preserving certification through contracts by validating that replacements fulfill contracted requirements. Prevelakis et al. [57] describe policies and mechanisms [29] for controlling resource access and communication in vehicles.

Future directions include cooperative access control similar to crypto currencies, and security-based trust in safety properties, confining compromised components with safety emerging from a quorum of proactively self-repairing units.

6.2 Fault and Attack Containment in Vehicular Networks

Arbitrary access of a compromised ECU to the bus, as mentioned before in the description of threat vector #3, can be prevented by a bus guardian [15], which restricts such accesses to the time window granted to the ECU. Techniques known from classical IT, such as Network Access Control, which restrict addition of unauthorized nodes, are worthwhile considering for in-car networks.

Since vehicle bus systems are of broadcast nature, intrusion detection mechanisms would have access to all traffic and could work on detecting anomalies. A main necessity of such an approach is the clear definition of patterns related to attacks on vehicle networks that could be translated as rules to the IDS [33]. An important challenge is to design these intrusion countermeasures to preserve the timing guarantees of bus accessing tasks while recovering compromised functionality (see Section 6.9 below).

In particular, it is not sufficient to encrypt messages to protect integrity and confidentiality (c.f., [80]). Given access to the bus, adversaries may alter the ciphertext of messages and thereby prevent their timely decryption. Timely delivery of correct information is a prerequisite for the timely operation of the control tasks, which depend on these messages. Redundant communication paths and dependability enhancing network coding [27] tolerate some localized attacks. Network coding can also be efficient against jamming attacks to wireless V2V and V2I networks [26]. An interesting question is how modern coding and fail-over techniques can help survive network-level attacks. The use of a rolling authentication window can ensure integrity and authenticity in vehicular networks [39].

6.3 Fault and Attack Containment in ECU and RSU Computing

For the more control oriented parts, AUTOSAR [1] is a prominent specification of software stacks with a partitioning kernel to guarantee temporal isolation between software components that for many aspects remain black boxes to the software integrators. AUTOSAR kernels (e.g., VxWorks [75], ErikaOS [24]) offer spatial and temporal isolation and an API for inter partition communication. Sysgo's PikeOS [69] offers among others an AUTOSAR personality on top of a separation microkernel. Future challenges include the integration and predictability of multicore [31, 72], GPGPUs [78, 36] and other heterogeneous computing elements such as FPGAs [30], which are increasingly required for complex autonomous driving applications such as sense and avoid. No protection is available against kernel compromise.

The security requirements discussed in the second threat vector also extend to the software installed in Road-Side Units. However, since these entities are stationary and an attractive target for adversaries, they have some specific requirements. A problem that needs special attention is how to prevent rogue RSUs, because a successful attack compromises an important point in the communications scheme of vehicular networks. A possible solution comprises the use of mutual authentication and authenticated software in RSUs. Intel TXT [28] offers remote attestation capabilities which an autonomous vehicle can use to authenticate the software executing on a RSU. This allows the vehicle to de-

Solutions (mechanisms, paradigms)	Vectors covered	Confidentiality	Integrity	Availability	Authenticity	Privacy	Non-Repudiation
Trust mgt. and access control	all *	*	*	*	*	*	*
Fault/attack containment in-car network	#3	x	✓	✓	✓	x	x
Fault/attack containment, ECU/RSU	#3, #4, #5	x	✓	✓	✓	x	x
Policy/trust enforcement	#1, #2, #3, #6	x	✓	✓	✓	x	✓
Security-aware safety-cases	#3, #4, #5, #6	x	✓	✓	x	x	✓
Privacy-preserving design/architecture	#1, #2, #6, #7	✓	x	x	x	✓	x
Supply-chain control	#4, #5, #8	x	✓	✓	x	x	x
Trusted-trustworthy roots-of-trust	#3, #4, #5, #7	✓	✓	✓	✓	✓	✓
Replication, diversity, self-healing	#4, #5, #7	x	✓	✓	x	x	x

Table 2. Threat vectors, solutions and their fulfillment of security properties.
(* - clear trust management and access control policies are prerequisites for subsequent solutions)

cide whether or not to trust the RSU based on the software it is executing, which prevents attackers from setting up a rogue RSU with malicious software. Intel's Software Guard Extensions (SGX) [21] build upon and extend TXT to avoid RSUs having to trust the RSU management operating system (OS). Besides offering authenticated boot functionality, SGX also offers trusted execution environments, which we discuss in Section 6.8. However, in the presence of an untrusted management OS, current versions of SGX cannot meet the availability guarantees cyber-physical systems require.

6.4 Policy and Trust Enforcement between Components

Depending on the sensitivity of data, vehicles might not need confidentiality or use encryption/decryption mechanisms in their communications, but they always need authentication in order to confirm that they are communicating with authenticated members of the network. For normal V2V communication, transmitters need only to hash messages and sign, while receivers verify those signatures. This procedure is relatively fast and fulfills the properties of integrity, authentication and non-repudiation. Sensitive data should be integrity and/or privacy protected prior to exposure in communication or storage (for further details, see also privacy-preserving design and architecture below).

Hamad et al. [29] propose a microkernel-based operating-system architecture for ECUs and demonstrate how kernel enforced capabilities can be combined with proxies to enforce local and global (i.e., cross ECU) communication policies. The security of their approach relies on a tamperproof microkernel, proxy and network stack for all in-car communication networks.

To prevent rogue RSUs and similar communication nodes in V2V networks, authentication must be coupled with communication as demonstrated by Parno [52]. Edge cloud solutions in or near RSUs are likely to run software components of multiple stakeholders without as clear a notion of responsibility as inside the car. For example, we expect driving behavior analyses by insurance companies, traffic management by public authorities responsible for the road segment and driver or manufacturer specific tasks offloading for example part of the autonomous driving stack. Whereas capabilities and similar access control mechanisms in the OS suffice to enforce communication policies regulating which component is authorized to communicate with which other component, additional mechanisms are required to control which information these components may reveal. Research in information-flow control (see Sabelfeld and Myers [61] for a survey) offers program analyses for preventing unauthorized disclosure of information. However, further research is required to better regulate partial release of privacy-sanitized data [35].

6.5 Security-aware Safety-case Procedures

Safety-case procedures for autonomous and connected cooperative vehicles require a change in mindset to not only accept defects and faults as stochastic events with possibly rare likelihoods of occurrence, but also as vulnerabilities that attackers may exploit systematically in a stochastics defying manner. Extending modern safety-analyses (e.g., [74]), we must equate accidental and malicious threats and analyze the likelihood of safety (catastrophic) failures under the perspective of both accidental and intentional causes. Security and safety must go hand in hand from the first hour and systems must be hardened by construction, in order to achieve resilience.

For in-vehicle systems, this implies not only considering ECUs, the mechanical systems they control and the software they run in isolation, but also their interplay with other ECUs in the system. For example, it is well known that exhaust management and engine control have a bidirectional physical coupling: the higher the speed, the more timing critical the engine control tasks become, and the higher the CO_2 concentration in the converter, which in turn increases the time-criticality of the injection control tasks regulating fuel injection and in turn CO_2 production.

Sophisticated combinations of attacks on the delicate timings in such coupled components, which would be minor and tolerable if accidental and/or in-isolation, may easily bring down the entire system in a non-detectable manner. Evidence of the difficulty to find timing errors can be derived from the work in [71], about an error in the timing analysis of CAN bus message delivery. Bril et al. [14] detected this error well over 10 years after industry adoption and long after this broken timing analysis supported the safety case of cars. Although it remained undetected, adversaries knowing about such a flaw might well have exploited it to bring down or take over the system, in a non-detectable manner.

6.6 Privacy-preserving Design and Architecture

Privacy concerns are a serious problem in the ecosystem under consideration because, as we have already mentioned, it handles personally identifiable information (PII). To try and prevent such problems we need to use *privacy-preserving* design strategies: privacy filters can be used to sanitize sensitive data (e.g., by removing faces from the car's video stream [46]); regularly updated pseudonyms may be used to sign messages without revealing the identity and thereby violating the privacy of their owners; and encryption may be used to communicate to outside infrastructures, such as the World Wide Web, or in private communications (e.g., between authorities or in emergency vehicles).

For example, Raya et al. [58] combine public key infrastructure (PKI) with pseudonyms to handle privacy in vehicular communication while maintaining a reasonable and stable performance with increasing network size. Sampigethaya et al. [62] propose group signatures for smaller environments such as platoons. Several authors [81, 45] link public identity information (e.g., name and vehicle identification number) to public keys in identity-based schemes.

6.7 Supply-chain Control

Supply-chain subversion has been heard of in other sectors, and is very difficult to counteract, given the large outsourcing nature of the car manufacturing business (see Section 6.8). A well-known example of fabrication-time attacks on hardware is the dopant-level Trojan, which converts benign circuitry into malicious circuitry through dopant ratio manipulation on input pins [13].

Researchers have proposed two basic approaches to defend against malicious hardware: (i) side-channel information such as power and temperature and (ii) additional sensors to measure features of a chip's behavior (e.g. signal propagation delay) [8, 44]. Although these defenses seem to work against large Trojans, they are not effective when considering recent attacks that require a single logic gate to change a CPU's mode [79].

Therefore, a possible approach in this area, is to counteract subversion ex-post, by using redundancy and diversity techniques. For example, by resorting to manycore architectures where it is possible to obtain core heterogeneity to limit the impact a particular manufacturer might have in the system's final security guarantees [12]. Using such a layout can permit the use of trusted hardware to monitor the behavior of untrusted hardware. These manycore solutions also rely on isolation at the network-on-chip level, which can overcome some of the known attacks against CPUs' privilege bits.

6.8 Trusted-trustworthy Root-of-trust Components

Critical components can, for example, rely on ultra-reliable hardware modules or microkernels to perform their functions. Intel's Software Guard Extensions (SGX) is a recent technology that provides isolated execution environments where security sensitive operations can be performed in a secure manner. For example, only authorized software executing in an isolated execution environment can handle control-sensitive data. This would prevent an attacker from manipulating the data to trick the image processing algorithms. Microkernels, on the other hand, are able to preserve availability despite untrusted management OSs, by virtualizing the latter [68].

Another opportunity for use of such components is for keeping the signing key of TAs secret. Because this key is the base of trust in the system, isolating and protecting it is crucial. They could be stored in extremely controlled physical environments with specialized security hardened hardware, such as Hardware Security Modules [76]. Researchers have also worked on providing the basic hardware components for the protection of vehicle communications, achieving confidentiality and integrity at the lower levels. Features include: an AES-128 bit engine in hardware, a True Random Number Generator, Hardware-shielded storage, secure system timer and a secure debugger [11].

Solutions for safe coordination between smart vehicles traveling in an uncertain environment have also been previously studied. The generic architecture uses architectural hybridization, separating components that execute baseline functionality in a predictable and certified way, like a safety kernel, from the generic payload components, which albeit supporting more powerful and versatile functionality, might be affected by run-time faults and uncertainties [16]. The system is guaranteed to switch from cooperative to baseline functionality when data validity is insufficient. This is a good approach to maintain safety even when, for example, communication channels are jammed.

6.9 Intrusion tolerance and self-healing

Availability is a must. Without it, cooperative schemes come to a halt and nodes are forced into a non-cooperative state. To mitigate this, redundant mechanisms or alternative communication media, such as GSM, should be available. Intrusion-resilient software techniques, such as secret sharing and proactive recovery [65, 32], have been used in other sectors, and we advocate their adoption in car systems. Byzantine fault tolerant solutions, together with proactive and reactive recovery techniques, can be deployed both in ECUs and in RSUs in order to achieve a resilient functionality [42, 66]. IT infrastructures based on different technologies have been proven to bring substantial security gains [25]. We expect the same benefits to follow in an autonomous and cooperative vehicle ecosystem. Because the structure of vehicles themselves varies, based on manufacturer, model and version, the road ecosystem is already quite diverse. Given their importance to the overall system, TAs and RSUs should apply similar techniques in order to enhance their intrusion resilience. MAFTIA [73] or COCA [82] are examples of secure, intrusion tolerant architectures, using this kind of techniques. Both frameworks aim at achieving high levels of availability while relying in secure back-ends for confidentiality and integrity. This is achieved with the combined use of threshold cryptography, proactive recovery and secure replication protocols, introduced above.

7. CONCLUSIONS

In this paper, we have studied the safety and security challenges impending on current and forthcoming vehicular systems, under the perspective of fully-fledged ecosystems of autonomous and cooperative vehicles. We started by arguing about why these systems will in a near future become important critical information infrastructures, simultaneously featuring connectivity, autonomy and cooperation, and why even today, threat analyses and safety cases should encompass both faults and attacks, the latter ranging from casual hackers to highly-skilled attackers.

By proposing to look holistically at this ecosystem of autonomous and cooperative vehicles, we hope to have made a contribution to a better understanding of the global threat plane and of the specific threat vectors designers should be attentive to. We acknowledge the contribution of prior literature cited, about specific vulnerabilities in sectors of this ecosystem, which can now be put in context with our global threat plane and vector analysis, to assess the likelihood and impact of successful attacks by agents with different power.

This study was highly biased toward car systems. However, the latter would perhaps be, from the security/safety binomial, the richer, most dynamic and most challenging of the ecosystems where autonomy and cooperation surface in several combinations. In consequence, we dare conjecture that many of the challenges encountered by other vehicle ecosystems, such as drones, for example, as well as some of the solutions, would fit in the present analysis. We leave a confirmation, and a possible generalization, for future work.

Further directions of future work include availability and timeliness of enclave-based security architectures, adaptive coding and failover for vehicular networks under attack and security-aware safety-case analysis, both with the aim to identify risks and to maintain known-safe operation in case of partial compromise.

8. REFERENCES

[1] *AUTOSAR: Automotive open system architecture.* Accessed: 2016-07-22.

[2] Media oriented systems transport. http://www.mostcooperation.com/.

[3] Motor vehicles increasingly vulnerable to remote exploits. https://www.ic3.gov/media/2016/160317.aspx. Accessed: 2016-03-21.

[4] Second tesla autopilot crash under federal scrutiny. http://money.cnn.com/2016/07/06/autos/tesla-autopilot-accident/. Accessed: 2016-07-08.

[5] Tesla's autopilot has had its first deadly crash. https://www.wired.com/2016/06/teslas-autopilot-first-deadly-crash/. Accessed: 2016-07-05.

[6] Google self-driving car project. Monthly Report, Feb. 2016.

[7] N. H. T. S. Administration et al. Preliminary statement of policy concerning automated vehicles. *Washington, DC*, 2013.

[8] D. Agrawal, S. Baktir, D. Karakoyunlu, P. Rohatgi, and B. Sunar. Trojan detection using ic fingerprinting. In *2007 IEEE Symposium on Security and Privacy (SP '07)*, pages 296–310, May 2007.

[9] M. S. Al-Kahtani. Survey on security attacks in vehicular ad hoc networks (VANETs). In *6th International Conference on Signal Processing and Communication Systems (ICSPCS)*, pages 1–9. IEEE, 2012.

[10] S. Al-Sultan, M. M. Al-Doori, A. H. Al-Bayatti, and H. Zedan. A comprehensive survey on vehicular ad hoc network. *Journal of network and computer applications*, 37:380–392, 2014.

[11] L. Apvrille, R. El Khayari, O. Henniger, Y. Roudier, H. Schweppe, H. Seudié, B. Weyl, and M. Wolf. Secure automotive on-board electronics network architecture. In *FISITA 2010 World Automotive Congress, Budapest, Hungary*, volume 8, 2010.

[12] N. Asmussen, M. Völp, B. Nöthen, H. Härtig, and G. Fettweis. M3: A hardware/operating-system co-design to tame heterogeneous manycores. In *21st Int. Conf. on Architectural Support for Programming Languages and Operating Systems*, ASPLOS '16, pages 189–203, New York, NY, USA, 2016. ACM.

[13] G. T. Becker, F. Regazzoni, C. Paar, and W. P. Burleson. Stealthy dopant-level hardware trojans. In *Proceedings of the 15th International Conference on Cryptographic Hardware and Embedded Systems*, CHES'13, pages 197–214, Berlin, Heidelberg, 2013. Springer-Verlag.

[14] R. Bril, J. Lukkien, R. Davis, and A. Burns. Message response time analysis for ideal controller area network (CAN) refuted. In *Proc. of the 5th International Workshop on Real-Time Networks (RTNâĂŹ06)*, 2006.

[15] G. Buja, A. Zuccollo, and J. Pimentel. Overcoming babbling-idiot failures in the FlexCAN architecture: a simple bus-guardian. In *10th IEEE conference on Emerging Technologies and Factory Automation (ETFA)*, volume 2, 2005.

[16] A. Casimiro, J. Kaiser, E. M. Schiller, P. Costa, J. Parizi, R. Johansson, and R. Librino. The Karyon project: Predictable and safe coordination in cooperative vehicular systems. In *43rd Annual IEEE/IFIP Conference on Dependable Systems and Networks Workshop (DSN-W)*, pages 1–12, 2013.

[17] A. Casimiro, J. Kaiser, and P. Verissimo. An architectural framework and a middleware for cooperating smart components. In *1st Conference on Computing Frontiers*, pages 28–39, Ischia, Italy, 2004. ACM.

[18] CBSNEWS. Toyota "unintended acceleration" has killed 89. http://www.cbsnews.com/news/toyota-unintended-acceleration-has-killed-89/. Accessed: 2016-07-22.

[19] S. Checkoway, D. McCoy, B. Kantor, D. Anderson, H. Shacham, S. Savage, K. Koscher, A. Czeskis, F. Roesner, T. Kohno, et al. Comprehensive experimental analyses of automotive attack surfaces. In *USENIX Security Symposium*. San Francisco, 2011.

[20] F. Consortium et al. Flexray communications system-protocol specification. *Version*, 2(1):198–207, 2005.

[21] V. Costan and S. Devadas. Intel SGX explained. Technical report, Massachusetts Institute of Technology, 2016. https://eprint.iacr.org/2016/086.pdf (Accessed: 2016-07-22).

[22] R. Di Pietro, S. Guarino, N. V. Verde, and J. Domingo-Ferrer. Security in wireless ad-hoc networks–a survey. *Comp. Comm.*, 51:1–20, 2014.

[23] J. R. Douceur. The Sybil attack. In *Peer-to-peer*

Systems, pages 251–260. Springer, 2002.

[24] E. Enterprise. Erika enterprise 3. http://erika.tuxfamily.org/drupal/content/erika-enterprise-3. Accessed: 2016-07-22.

[25] M. Garcia, A. Bessani, I. Gashi, N. Neves, and R. Obelheiro. OS diversity for intrusion tolerance: Myth or reality? In *IEEE/IFIP 41st International Conference on Dependable Systems & Networks (DSN)*, pages 383–394, 2011.

[26] M. Ghaderi, D. Goeckel, A. Orda, and M. Dehghan. Efficient wireless security through jamming, coding and routing. In *Sensor, Mesh and Ad Hoc Communications and Networks (SECON)*, New Orleans, USA, June 2013. IEEE.

[27] M. Ghaderi, D. Towsley, and J. Kurose. Reliability gain of network coding in lossy wireless networks. In *IEEE INFOCOM*, Phoenix, USA, April 2008.

[28] J. Greene. *Intel Trusted Execution Technology — Hardware-based Technology for Enhancing Server Platform Security*, 2010. http://www.intel.de/content/dam/www/public/us/en/documents/white-papers/trusted-execution-technology-security-paper.pdf (Accessed: 2016-07-22).

[29] M. Hamad, J. Schlatow, V. Prevelakis, and R. Ernst. A communication framework for distributed access control in microkernel-based systems. In *12th Annual Workshop on Operating Systems Platforms for Embedded Real-Time Applications (OSPERT16)*, pages 11–16, Toulouse, France, July 2016.

[30] J. Hauswald, M. A. Laurenzano, Y. Zhang, C. Li, A. Rovinski, A. Khurana, R. G. Dreslinski, T. Mudge, V. Petrucci, L. Tang, et al. Sirius: An open end-to-end voice and vision personal assistant and its implications for future warehouse scale computers. In *Architectural Support for Programming Languages and Operating Systems (ASPLOS)*, pages 223–238. ACM, 2015.

[31] J. L. Herman, C. J. Kenna, M. S. Mollison, J. H. Anderson, and D. M. Johnson. Rtos support for multicore mixed-criticality systems. In *18th IEEE Real Time and Embedded Technology and Applications Symposium*, pages 197–208, April 2012.

[32] A. Herzberg, S. Jarecki, H. Krawczyk, and M. Yung. Proactive secret sharing or: How to cope with perpetual leakage. In *Advances in Cryptology - CRYPT0 95*, pages 339–352. Springer, 1995.

[33] T. Hoppe, S. Kiltz, and J. Dittmann. Security threats to automotive CAN networks–practical examples and selected short-term countermeasures. In *Computer Safety, Reliability, and Security*, pages 235–248. Springer, 2008.

[34] Y.-C. Hu, A. Perrig, and D. B. Johnson. Wormhole attacks in wireless networks. *IEEE Jour. on Selected Areas in Communications*, 24(2):370–380, 2006.

[35] J.-P. Hubaux and A. Juels. Privacy is dead, long live privacy. *Commun. ACM*, 59(6):39–41, May 2016.

[36] S. Kato, K. Lakshmanan, R. Rajkumar, and Y. Ishikawa. Timegraph: Gpu scheduling for real-time multi-tasking environments. In *USENIX Annual Technical Conference (ATC)*. USENIX, 2011.

[37] S. Khandelwal and A. Abhale. Topology base routing attacks in vehicular ad hoc network — survey. *International Journal of Advanced Research in Computer Science and Software Engineering*, 3(11), Nov. 2013.

[38] R. Kirk. Cars of the future: the Internet of Things in the automotive industry. *Network Security*, 2015(9):16–18, 2015.

[39] P. Koopman and C. Szilagyi. Integrity in embedded control networks. *IEEE Security & Privacy*, 11(3):61–63, 2013.

[40] H. Kopetz and P. Veríssimo. *Distributed Systems*, chapter Real-Time and Dependability Concepts. ACM-Press, Addison-Wesley, 2nd edition, 1993.

[41] K. Koscher, A. Czeskis, F. Roesner, S. Patel, T. Kohno, S. Checkoway, D. McCoy, B. Kantor, D. Anderson, H. Shacham, et al. Experimental security analysis of a modern automobile. In *IEEE Symposium on Security and Privacy (SP)*, pages 447–462, 2010.

[42] L. Lamport, R. Shostak, and M. Pease. The Byzantine generals problem. *ACM Trans. on Progr. Languages and Systems*, 4(3):382–401, 1982.

[43] J. Lewis. A smart bomb in every garage? Driverless cars and the future of terrorist attacks. https://www.start.umd.edu/news/smart-bomb-every-garage-driverless-cars-and-future-terrorist-attacks. Accessed: 2016-03-21.

[44] J. Li and J. Lach. At-speed delay characterization for ic authentication and trojan horse detection. In *Hardware-Oriented Security and Trust, 2008. HOST 2008. IEEE International Workshop on*, pages 8–14, June 2008.

[45] H. Lu, J. Li, and M. Guizani. A novel ID-based authentication framework with adaptive privacy preservation for vanets. In *Computing, Communications and Applications Conference (ComComAp)*, pages 345–350. IEEE, 2012.

[46] P. Ludivig. Detection of faces for mobile robots. Master's thesis, Faculty of Science, Technology and Communication - University of Luxembourg, Aug. 2015. Sec. 5.6: Face Pixelation.

[47] M. N. Mejri, J. Ben-Othman, and M. Hamdi. Survey on VANET security challenges and possible cryptographic solutions. *Vehicular Communications*, 1(2):53–66, 2014.

[48] C. Miller and C. Valasek. Remote exploitation of an unaltered passenger vehicle. *Black Hat USA*, 2015.

[49] B. Mokhtar and M. Azab. Survey on security issues in vehicular ad hoc networks. *Alexandria Engineering Journal*, 54(4):1115–1126, 2015.

[50] J. Newsome, E. Shi, D. Song, and A. Perrig. The Sybil attack in sensor networks: analysis & defenses. In *3rd International Symposium on Information processing in sensor networks*, pages 259–268. ACM, 2004.

[51] C. of Oklahoma County. Bookout vs toyota. http://www.safetyresearch.net/Library/Bookout_v_Toyota_Barr_REDACTED.pdf. Accessed: 2016-07-22.

[52] B. Parno. Bootstrapping trust in a "trusted" platform. In *Proceedings of the 3rd Conference on Hot Topics in Security*, HOTSEC'08, pages 9:1–9:6, Berkeley, CA, USA, 2008. USENIX Association.

[53] K. Pazul. Controller area network (CAN) basics. *Microchip Techn. Inc*, 1999.

[54] D. Perez and J. Pico. A practical attack against GPRS/EDGE/UMTS/ HSPA mobile data communications. *Black Hat DC*, 2011.

[55] J. Petit and S. E. Shladover. Potential cyberattacks on automated vehicles. *IEEE Trans. on Intelligent Transportation Systems*, 16(2):546–556, 2015.

[56] J. Petit, B. Stottelaar, M. Feiri, and F. Kargl. Remote attacks on automated vehicles sensors: Experiments on camera and lidar. In *Black Hat Europe*, 2015.

[57] V. Prevelakis and M. Hammad. A policy-based communications architecture for vehicles. In *International Conference on Information Systems Security and Privacy*, Angers, France, Feb. 2015.

[58] M. Raya and J.-P. Hubaux. Securing vehicular ad hoc networks. *Journal of Computer Security*, 15(1):39–68, 2007.

[59] M. Ruff. Evolution of local interconnect network (LIN) solutions. In *58th Vehicular Tech. Conf. (VTC 2003-Fall)*, volume 5, pages 3382–3389. IEEE, 2003.

[60] J. Rufino, N. Pedrosa, J. Monteiro, P. Verissimo, and G. Arroz. Hardware support for can fault-tolerant communication. In *Proceedings of the 5th IEEE International Conference on Electronics, Circuits and Systems, Lisboa, Portugal, September 1998*, Sept. 1998.

[61] A. Sabelfeld and A. C. Myers. Language-based information-flow security. *IEEE J.Sel. A. Commun.*, 21(1):5–19, Sept. 2006.

[62] K. Sampigethaya, L. Huang, M. Li, R. Poovendran, K. Matsuura, and K. Sezaki. CARAVAN: Providing location privacy for VANET. Technical report, DTIC Document, 2005.

[63] J. Schlatow, R. Ernst, M. Nolte, and M. Maurer. Contract-based automated integration for complex component-based systems. Design, Automation and Test in Europe - Demo, March 2016.

[64] J. Schlatow, M. Moestl, and R. Ernst. An extensible autonomous reconfiguration framework for complex component-based embedded systems. In *IEEE International Conference on Autonomic Computing (ICAC)*, pages 239–242, July 2015.

[65] A. Shamir. How to share a secret. *Comm. of the ACM*, 22(11):612–613, 1979.

[66] P. Sousa, A. N. Bessani, M. Correia, N. F. Neves, and P. Verissimo. Highly available intrusion-tolerant services with proactive-reactive recovery. *IEEE Trans. on Parallel & Distributed Systems*, pages 452–465, 2009.

[67] J. Staggs. How to hack your mini cooper: Reverse engineering CAN messages on passenger automobiles. *Institute for Information Security*, 2013.

[68] F. Stumpf, C. Meves, B. Weyl, and M. Wolf. A security architecture for multipurpose ECUs in vehicles. In *25th Joint VDI/VW Automotive Security Conference*, Ingolstadt, Germany, 2009.

[69] Sysgo. Pikeos. https://www.sysgo.com/solutions/industry-solutions/automotive/. Accessed: 2016-07-22.

[70] C. Szegedy, W. Zaremba, I. Sutskever, J. Bruna, D. Erhan, I. Goodfellow, and R. Fergus. Intriguing properties of neural networks. *arXiv preprint arXiv:1312.6199*, 2013.

[71] K. Tindell, H. Hansson, and A. J. Wellings. Analysing real-time communications: Controller area network (CAN). In *Proc. of the 15th Real-Time Systems Symposium (RTSS'94)*, 1994.

[72] T. Ungerer, F. Cazorla, P. Sainrat, G. Bernat, Z. Petrov, C. Rochange, E. Quinones, M. Gerdes, M. Paolieri, J. Wolf, H. Casse, S. Uhrig, I. Guliashvili, M. Houston, F. Kluge, S. Metzlaff, and J. Mische. Merasa: Multicore Execution of Hard Real-Time Applications Supporting Analyzability. *IEEE Micro*, 30(5):66–75, 2010.

[73] P. E. Verissimo, N. F. Neves, C. Cachin, J. Poritz, D. Powell, Y. Deswarte, R. Stroud, and I. Welch. Intrusion-tolerant middleware: the road to automatic security. *IEEE Security Privacy*, 4(4):54–62, July 2006.

[74] E. Villani, N. Fathollahnejad, R. Pathan, R. Barbosa, and J. Karlsson. Reliability analysis of consensus in cooperative transport systems. In M. Roy, editor, *SAFECOMP 2013 - Workshop ASCoMS (Architecting Safety in Collaborative Mobile Systems) of the 32nd International Conference on Computer Safety, Reliability and Security*, Toulouse, France, Sept. 2013.

[75] Windriver. Vxworks. http://windriver.com/products/vxworks/. Accessed: 2016-07-22.

[76] M. Wolf and T. Gendrullis. Design, implementation, and evaluation of a vehicular hardware security module. In *Information Security and Cryptology-ICISC 2011*, pages 302–318. Springer, 2011.

[77] M. Wolf, A. Weimerskirch, and C. Paar. Secure in-vehicle communication. In *Embedded Security in Cars*, pages 95–109. Springer, 2006.

[78] Y. Xu, R. Wang, T. Li, M. Song, L. Gao, Z. Luan, and D. Qian. Scheduling tasks with mixed timing constraints in gpu-powered real-time systems. In *Proceedings of the 2016 International Conference on Supercomputing*, page 30. ACM, 2016.

[79] K. Yang, M. Hicks, Q. Dong, T. Austin, and D. Sylvester. A2: Analog malicious hardware. In *IEEE Security and Privacy (SP)*, 2016.

[80] J. Yoshida. Can bus can be encrypted, says trillium. http://www.eetimes.com/document.asp?doc_id=1328081, Oct. 2015. Accessed: 2016-07-22.

[81] K. Zaidi, Y. Rahulamathavan, and M. Rajarajan. Diva-digital identity in vanets: A multi-authority framework for vanets. In *19th IEEE International Conference on Networks (ICON)*, pages 1–6, 2013.

[82] L. Zhou, F. B. Schneider, and R. Van Renesse. COCA: A secure distributed online certification authority. *ACM Transactions on Computer Systems (TOCS)*, 20(4):329–368, 2002.