Intrusion Resilience Systems for Modern Vehicles Position Paper

Ali Shoker, Paulo Esteves-Verissimo

Intrusion Resilience Systems for Modern Vehicles

Position Paper

Ali Shoker and Paulo Esteves-Verissimo
Resilient Computing and Cybersecurity Center (RC3),
Computer, Electrical and Mathematical Sciences and Engineering Division (CEMSE),
King Abdullah University of Science and Technology (KAUST)
Thuwal 23955-6900, Kingdom of Saudi Arabia
{ali.shoker, paulo.verissimo}@kaust.edu.sa

***Abstract*—We introduce the concept of Intrusion Resilience Systems (IRS) for modern vehicles. An IRS is a middleware that enables running a vehicular application in a replicated way, i.e., as a *Replicated State Machine*, over several *ECU*s. By requiring the replicated processes to reach a form of *Byzantine* agreement before changing their local state, the IRS ensures the resilience of critical vehicular applications despite assumed faults or attacks, as long as threat assumptions are met. This position paper proposes the tentative architecture of IRS and discusses its conceptual feasibility and underlying challenges. Our study rides the mutation of modern vehicular environments, which are closing the gap between simple and resource-scarce 'real-time and embedded systems', and complex and powerful 'information technology' ones. We show that current architectures are becoming plausible for such modular fault and intrusion tolerance solutions—deemed too heavy in the past. Our conclusion is that this topic deserves more attention in both academia and industry.**

***Index Terms*—modern vehicles, intrusion resilience, cybersecurity, Byzantine agreement**

I. Introduction

Three trends, Automation, Digitization, and Connectivity are disrupting the ways modern vehicles are designed and used. While these trends can bring notable features like safety, efficiency, and convenience, they could turn into a curse if security and resilience are left as afterthoughts. Even "safety" [4], [15], more recently, risks being sacrificed to the altar of ML-fueled vision recognition [25], [28]. Unfortunately, reality shows that safety and security incidents are doubling annually during the past three years, causing up to half Trillion dollars by 2024 due to cyberattacks [19], and leading to millions of car recalls [7]. Such trend, if not contradicted, jeopardizes the sought features and puts human lives at risk. We need novel approaches to improve vehicles' resilience: ensuring that an acceptable service prevails, even in uncertain environment conditions, or in the presence of faults or attacks that might not have been predicted.

This work is motivated by two main observations in the automotive industry. The first is that the automation and digitization trends increase the complexity of vehicles and the likelihood of software faults and vulnerabilities. While digitization suggests software-defined vehicle systems (compute nodes, networks, and software) as a main enabler to automation, these systems must be trustworthy since automation implicitly delegates the vehicle's driver control to the automotive system, e.g., supporting features like x-by-wire, Advanced Driver Assistance Systems (ADAS), and Telematics. To this end, digitization involved a considerable number of distributed software components running on over a hundred embedded compute devices, *Electronic Control Unit*s (ECU), which communicate via in-vehicle digital networks, e.g., CAN bus, Automotive Ethernet, FlexRay, etc. [20]. This results in a complex system with an enormous number—estimated to exceed 100 Millions—of Software Lines of Code (SLoC) in mainstream vehicles [7], [22]. Experience shows that human errors are positively correlated with both system's complexity and code footprint, and this increases the likelihood of benign faults and intrusions.

The second observation is that since digitization is closing the gap with the Information Technology (IT) and Internet of Things (IoT), it is imperative to connect the vehicle to its digital counterparts in the cyberspace. Connectivity is established in several networking forms like Vehicle to Everything (V2X), Cellular, 5G, Bluetooth, WIFI, GPS, or even through hardware memory sticks or USB connectivity [8]. This raises substantial security challenges as it boosts the attack surface and entry points of the vehicle system and makes it highly prone to intrusions induced by (the well experienced) attackers in the cyberspace, via exploiting the existing vulnerabilities [18], [31] (discussed in the first observation).

The automotive community has been recently focusing on consolidating the network security layer, leaving the higher software layers insufficiently addressed. Of particular interest is the introduction of new network security controls and tools (e.g., Gateways, Firewalls), and hardening the security of existing networks, e.g., CAN Bus, CAN FD, CAN XL, FlexRay, Automotive Ethernet (100BASE-T1 and 10BASE-T1s), etc. [20]. This is also supported by using endpoint tools like *Intrusion Detection Systems* (IDS) and *Intrusion Prevention Systems* (IPS) [14]. IDS systems of either "school"—signature-based and anomaly-based IDS—have limitations in the context of in-car systems, respectively blindness to zero-day vulnerabilities, and being difficult to define a "normal behavior". Not to mention the problem of reaction/mitigation in real-time, which haunts IPS, and makes these ad-hoc response techniques currently very limited (e.g., detaching a vulnerable ECU from the network bus using the Bus-off state [14], [21]). On the other hand, since the network PHY/MAC protocols and tools (IPS/IDS) are application-agnostic, they can neither

detect the anomalies and intrusions occurring at the upper layers nor stop their propagation to other ECUs.

In this position paper, we introduce the concept of *Intrusion Resilience Systems* (IRS) for modern vehicles. IRS aims at contributing to a timely revolution in current in-vehicle computer and network architectures, by extending the security and safety properties of component-based architectures (e.g., AUTOSAR). We propose SW-implemented fault and intrusion tolerance, leveraging available sets of failure-independent ECUs, similarly to the principles laid down by IT pioneering architectures of the 80's in *modular, incremental fault-tolerance* [3], [23]. The approach is inline with the increasing demand for automotive computing and network channel redundancy, i.e., *ASIL Decomposition*, as part of the ISO 26262v.2 safety standard [12], [13].

The novel concepts behind IRS system-level automotive middleware allow running multiple and possibly diverse replicas of a state-full application process on different ECUs, forming a resilient deterministic *Replicated State Machine* [27]. Replicas are required to agree on a common state through a variant of *Byzantine Agreement* [5] protocols (today widely used in *Blockchain*) prior to changing their local state. As long as the process is deterministic, agreement is reached despite the existence of benign or intrusion faults in a minority of replicas. Distributed applications like door locks, window control, software Over-the-Air (OTA) updates are few examples on feasible applications on top of IRS.

IRS gives a quantum leap from IDS/IPS functions. First, it can work at a higher level of abstraction, targeting application software level anomalies and intrusions. Second, IRS follows an error masking approach which virtually captures all faults, even unknown ones, unlike IDS systems. Third, contrary to IPS whose response often degrades or suspends some system components or functions [14], [21], IRS makes it possible to roughly maintain the application functionality and quality under failures or attack.

In this work, we present a tentative IRS architecture to demonstrate the our concept; and we then drive a logical reasoning for the feasibility of IRS for vehicles. Therefore, we analyze the technological advancements in modern vehicles, including applications, distributed architectures, ECUs with decent computational and storage capacity, and improved in-vehicle networks. An empirical evaluation system meeting these different networking, architecture and application requirements is a work in progress in our team.

II. Intrusion Resilience System

A. Systems and Threat Models

Consider an in-vehicle system of N *nodes*. A *node* is composed of an computing device, i.e., an ECU, a corresponding software stack, and a (critical) soft real-time vehicular application for simplicity. (This can be generalized to many applications.) A node can communicate with its counterparts through messaging via a vehicular *network*, either through a direct link, a switch, or via a gateway. A sent *message* is assumed to eventually reach its destination node despite network failures or attacks (e.g., after re-transmissions). A node has a unique identity in the system to verify message authenticity and integrity using lightweight cryptography primitives, like *Elliptic Curve Cryptography* (ECC). A node, or the application therein, is assumed to be *deterministic*. However, an application can fail by crashing or behave arbitrarily or maliciously when subject to an intrusion. We assume that at most a fraction F of N nodes can fail at a time, which implicitly assumes some independence of failures between nodes. This can be achieved by employing ECUs from diverse vendors, different libraries, software stack, and implementation, etc., which is not uncommon in the automotive setting. Finally, we assume the existence of a technique to detect *Denial of Service* (DoS) jamming attack in multi-hop bus networks like CAN and 10BASE-T1s [14], [20].

B. Architecture and Concept

a) Concept: The IRS concept is based on the idea of intrusion error *masking* rather than detection and prevention as in IPS/IDS. By running multiple (N) replicas/versions of an application and comparing their outputs on different nodes (ECUs), it is possible to mask any error caused by accidental or malicious faults occurring on F faulty nodes, by adopting the output state of an uninfected majority ($N - F$). This is possible through running a Byzantine agreement protocol across application replicas. In this approach, the state of a critical application can only be modified upon the agreement of at least $N-F$ counterparts. This exploits the current replicated vehicle functionalities, often used for coordinated actuation and notification, to improve intrusion resilience.

b) Architecture: We present the IRS system view architecture in Fig. 1, A. The System View shows a number N of IRS nodes ($N = 4$, in this case) replicated over N ECUs. For clarity, we use *Zonal Control Units (ZCU)* as ECUs to host different applications (e.g., door locks and window control) on the same ECU. On the other hand, Fig. 1, B presents the Node View at one of the nodes (i.e., node 2) describing its components and relation within the Hardware/Software (HW/SW) stack.

In particular, the IRS stands as a middleware or service used by those critical applications that require intrusion resilience. N versions of the application are employed over N different nodes, making use of the underlying IRS middleware. The core module of the IRS seeks to ensure *agreement* on requests issued by the application via an *IRS proxy*. The proxy encapsulates the authentication, peer information, and the function to be made resilient through IRS in an application-agnostic way. The agreement module runs the main Byzantine agreement protocol to ensure (1) *total ordering* on the application state and (2) output validation (i.e., comparison of results from counterpart nodes on other ECUs). The agreement module benefits from three underlying modules, namely, *Discovery*, *Broadcast*, and *Overlay* to facilitate the membership management and networking with the peer nodes as a separate layer. Note that IRS can make use of these modules if made available by other frameworks, e.g., in the AutoSAR architecture.

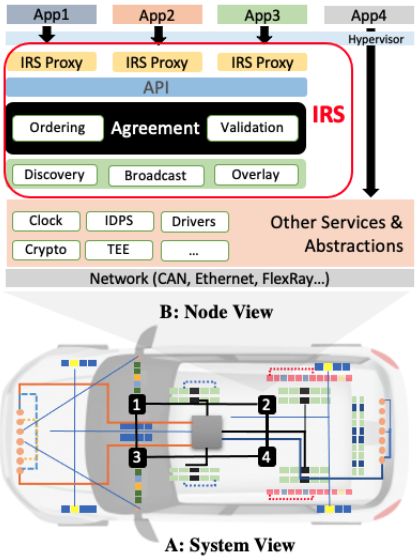

Fig. 1. Intrusion Resilience System (IRS) Architecture.

IRS offers *modular and incremental fault and intrusion tolerance* [23]. Not all node applications—or even functions of an application—are supposed to use the IRS, as they might not be critical, e.g., the case of App4 in the figure. Likewise, applications using IRS may resort to different models of replication (from crash to Byzantine fault tolerance), as well as different sizes of tolerance quorums (#(N)). For instance, an application that controls the remote door locks is much more critical than the mirror tilting application. Similarly, an Over-the-Air (OTA) update application is highly critical compared to infotainment social network (e.g., chatting) update.

IRS runs on top of other basic services and abstractions, such as those defined in the AutoSAR standard[1]. This way, it facilitates the integration of resilience in the existing component-based automotive architecture philosophy. At this layer, other tools like IDS, IPS may operate as well. Finally, the bottom layer encapsulates the PHY network protocols (e.g., CAN, FlexRay, Automotive Ethernet) typically managed by the physical controller. ECUs are connected via a network that could be multidrop, node-to-node, or switch-based network as long messages sent by one node are eventually delivered at the destination node.

c) Byzantine Agreement: IRS encapsulates a distributed voting logic using an intrusion tolerant protocol category based on the concept of Byzantine Agreement/Consensus. Initial practical protocols [5] would require $N = 3F + 1$, had quadratic ($O(n^2)$) messaging complexity, and were computationally demanding due to the heavy use of cryptography. The following generation was architecturally hybrid [2], featuring the use of trusted-trustworthy components [9], [32], dramatically reducing complexity, and requiring a smaller quorum of $N = 2F + 1$. Later, the advent of *Blockchain* inspired yet another generation of intrusion tolerant protocols, becoming even more efficient and lightweight [10], [33]. The current state of affairs makes them feasible for environments with moderate capacities like modern vehicles (more on this in the next section). Describing a specific protocol is out of the scope of this position paper; however, we are currently validating two variants of Byzantine Resilient Real-Time protocols, namely, *RTByzCast* and *PISTIS* [16], [17]. Unlike previous intrusion tolerant protocols, which were non-synchronous, these real-time protocols are suited for hard or soft real-time environments as they have *Timeliness* properties to guarantee delivery/execution given a defined probabilistic time-bound. The main challenges we envision are validating the *Safety* and *Liveness* properties in the Byzantine threat model over in-vehicle networks, and the behavior with multi-drop (broadcast) networks and their synchrony models.

[1]https://www.autosar.org/

III. Feasibility Discussion

While the need for building resilient systems is very well understood, applying redundancy-based solutions like IRS may look infeasible for in-vehicular systems. Nevertheless, we argue that this is no longer the case as the three trends automation, digitization, and connectivity have changed modern vehicular systems dramatically. In this section, we try to alleviate these concerns by driving a conceptual analysis demonstrating the potential feasibility of IRS to modern vehicles.

A. Distributed and Redundant Applications

The current application landscape in automotive is very rich and complex, spanning ADAS & Safety Systems, Infotainment, Body Electronics, Powertrain, and Telematics. At a fine-grained level, these applications incur millions of functionalities. For instance, a Volvo modern vehicle "contains 10 million conditional statements as well as 3 million functions, which are invoked some 30 million places in the source code" [7]. Many of these applications are becoming naturally distributed across the vehicle to manage the dependencies between functionalities and to synchronize the similar ones across the vehicle. For instance, a vehicle may have applications running four steering, braking, tyre pressure processes; four/five door lock and window processes; four light sets of processes, two mirror processes, several airbag processes, etc. Nevertheless, these processes are currently only synchronized in a passive way, i.e., propagating notifications, where "decisions", e.g., changing an actuator state, are only made locally. Given this, the overhead of enforcing distributed control through agreement protocols prior to changing the application state would be reasonably low since replicas are already being used. This is sound for safety/security critical applications that are soft-real time, in particular, like door lock/unlock, window open/close, and OTA update validation by different processes on different ECUs.

On the other hand, using redundancy to boost vehicle safety is becoming increasingly required [13]. Indeed, the *ASIL Decomposition* mechanism drafted in the ISO 26262v.2 automotive safety standard [12], [13] suggests using redundant computing nodes and network channels to improve safety

and reduce the costs (e.g., by using redundant cheaper nodes instead of one expensive node).

B. Distributed Architecture

The vehicular architecture has become heavily distributed as more ECUs are being added over time to cope with the application demands. Considering the evolution of distributed architectures [22], applications are becoming more aggregated in larger ECUs: (1) Domain-based ones aggregate applications with similar functionalities; (2) Zonal-based ones aggregate based on the vehicle zone, e.g., a Door Control Unit hosts many applications (like door locks, motors, windows, theme lights) at the door proximity; and (3) Centralized. The former two are considered very convenient environments to run replicated protocols as the agreement protocol suggested in IRS. In addition, multiple aggregated applications can directly benefit from the IRS being a middleware/service. Indeed, while the replication cost has always been an adoption barrier in the IT world, the costs (surprisingly) look lower in vehicular architectures being natively distributed.

The latter centralized architecture is getting more traction recently. We do not recommend this architecture from a security perspective, being a single point of failure/attack. Nevertheless, transforming the central controller into a distributed cluster could be a trade-off solution to mitigate this risk significantly.

C. Efficient and Secure Networks

Vehicular networks, especially the CAN bus, have always been considered slow and the weakest spot in a vehicle. In particular, the classical baud rate of CAN bus cannot be higher than 1Mbps, and the payload is only 8 bytes per packet[2]. This prohibits an IRS-like solution where the agreement meta-data size (identifiers, signatures, cryptographic digests, clock) is high. On the other hand, CAN frames lack the sender/receiver identifiers which makes authentication and integrity a non-trivial task. However, the new versions of CAN, i.e., CAN FD and XL, have larger frame's payload size of 64B and 2KB, and baud rate to 2Mbps and 10Mbps, respectively. These are considered acceptable for soft real-time applications, e.g., like door locks and OTA updates, as response time is not critical. Furthermore, novel networks like Automotive Ethernet and FlexRay have native security support and an order of magnitude higher bit rate. We believe that these advancements mitigate the concerns regarding the feasibility of IRS to such environment.

D. Decent HW/SW Stack

It can be assumed that running IRS agreement protocols in a constrained device (like a micro-controller-based ECU) and networks would be an overkill due to the heavy use of cryptography. Despite being challenging, modern automotive ECUs (microprocessor-based and multi-core) are getting high computational and storage capacities that could be compared to a *Raspberry Pi* or a mobile phone[3]. This is correct, in particular, for main ECUs like domain and zone controllers, gateways, telecommunication units, etc. On top of this hardware, the software stack [26] is also getting more mature while we observe more UNIX, POSIX, and Linux-based RTOS/OS, e.g., AGL, RTLinux, QNX, Android Auto, and Apple CarPlay. This also means that a lot of IT/IoT libraries could now be adapted or used in automotive. New architectures are widely adopting the virtualization hypervisor technology, which facilitates application deployments on an ECU, and thus, replication in our case [26]. Therefore, the modern HW/SW stack of modern vehicles is decent enough to support a solution like IRS.

Furthermore, independence of failures between IRS replicas is considered a main challenge, being key for the effectiveness in common-mode vulnerabilities or faults. The rich supply chain of automotive HW/SW is useful to generate diversity, which is the main approach to improve independence of failures, among others [1], [6], [11], [24], [29], [30]. For instance, it not uncommon to have ECUs or MCUs of the same specifications from different vendors; diverse software libraries, operating systems and hypervisors. Even at application level, one can choose only the critical functions to run over IRS, which will require only these functions to be implemented by different teams, e.g., using *N-version programming*.

IV. Conclusion

We introduced the concept of *Intrusion Resilience Systems* (IRS) for modern vehicles. The aim is to bridge the gap left in security-by-design and intrusion detection and prevention systems at two levels: first, it is tailored for the software/application layer; second, it tolerates faults and intrusions to roughly maintain the same service quality even if intrusions could not be profiled. IRS uses the *State Machine Replication* approach in which the replicated application can only change the local state upon *Byzantine agreement* with its counterpart nodes. The paper proposed a preliminary architecture and an analytic feasibility study that highlights the fact that modern vehicular technologies are closing the gap with IT/IoT technologies, which makes them plausible environments to adopt a replicated solution as IRS. We believe that our approach is therefore worth consideration, and we encourage researchers and practitioners to investigate this direction by studying the tradeoffs of agreement protocols, architectures, diversity, application space, etc.

As a work progress, we are currently investigating the feasibility of IRS over CAN and Automotive Ethernet variants to validate the application timing constrains and bandwidth efficiency. Primary results indicate that under malicious threat models, Automotive Ethernet and FlexRay are more suited than CAN (and any multi-hop network, in general), since the latter is more prone to jamming and spoofing attacks.

[2]The CAN bus is a broadcast-based network that arbitrates message identifiers on a bit basis. The lower the identifier (composed of a number of bits, e.g., 11 bits) the highest its priority. In this sense, a competing message with lowest identifier can take over sending over the bus. This imposes limits on payload sizes to maintain high message frequencies.

[3]https://www.emobility-engineering.com/focus-ecus/

REFERENCES

[1] Benoit Baudry and Martin Monperrus. The multiple facets of software diversity: Recent developments in year 2000 and beyond. *ACM Computing Surveys (CSUR)*, 48(1):1–26, 2015.

[2] Johannes Behl, Tobias Distler, and Rüdiger Kapitza. Hybrids on steroids: Sgx-based high performance bft. In *Proceedings of the Twelfth European Conference on Computer Systems*, pages 222–237, 2017.

[3] Kenneth P Birman. Isis: A system for fault-tolerant distributed computing. Technical report, CORNELL UNIV ITHACA NY DEPT OF COMPUTER SCIENCE, 1986.

[4] António Casimiro, Jörg Kaiser, Elad M Schiller, Pedro Costa, José Parizi, Rolf Johansson, and Renato Librino. The karyon project: Predictable and safe coordination in cooperative vehicular systems. In *2013 43rd Annual IEEE/IFIP Conference on Dependable Systems and Networks Workshop (DSN-W)*, pages 1–12. IEEE, 2013.

[5] Miguel Castro, Barbara Liskov, et al. Practical byzantine fault tolerance. In *OsDI*, volume 99, pages 173–186, 1999.

[6] Miguel Castro, Rodrigo Rodrigues, and Barbara Liskov. Base: Using abstraction to improve fault tolerance. *ACM Transactions on Computer Systems (TOCS)*, 21(3):236–269, 2003.

[7] Robert N. Charette. How software is eating the car. 2021.

[8] Riccardo Coppola and Maurizio Morisio. Connected car: technologies, issues, future trends. *ACM Computing Surveys (CSUR)*, 49(3):1–36, 2016.

[9] Miguel Correia, Nuno Ferreira Neves, and Paulo Verissimo. Bft-to: Intrusion tolerance with less replicas. *The Computer Journal*, 56(6):693–715, 2013.

[10] Jérémie Decouchant, David Kozhaya, Vincent Rahli, and Jiangshan Yu. Damysus: Streamlined bft consensus leveraging trusted components. In *Proceedings of the Seventeenth European Conference on Computer Systems*, EuroSys '22, page 1–16, New York, NY, USA, 2022. Association for Computing Machinery.

[11] Miguel Garcia, Alysson Bessani, Ilir Gashi, Nuno Neves, and Rafael Obelheiro. Analysis of operating system diversity for intrusion tolerance. *Software: Practice and Experience*, 44(6):735–770, 2014.

[12] International Organization for Standardization. ISO 26262-1:2018 - Road vehicles — Functional safety. https://www.iso.org/standard/68383.html, 2018. Online; accessed Sep, 2022.

[13] Jitin George. C2000™ MCU SafeTI control solutions: An introduction to ASIL decomposition and SIL synthesis. https://www.ti.com/lit/fs/sway028/sway028.pdf?ts=1662361210895ref$_{u}rl = https2019.Online; accessedSep$, 2022.

[14] Kyounggon Kim, Jun Seok Kim, Seonghoon Jeong, Jo-Hee Park, and Huy Kang Kim. Cybersecurity for autonomous vehicles: Review of attacks and defense. *Computers Security*, 103:102150, 2021.

[15] Philip Koopman and Michael Wagner. Autonomous vehicle safety: An interdisciplinary challenge. *IEEE Intelligent Transportation Systems Magazine*, 9(1):90–96, 2017.

[16] David Kozhaya, Jérémie Decouchant, and Paulo Esteves-Verissimo. Rt-byzcast: Byzantine-resilient real-time reliable broadcast. *IEEE Transactions on Computers*, 68(3):440–454, 2018.

[17] David Kozhaya, Jérémie Decouchant, Vincent Rahli, and Paulo Esteves-Verissimo. Pistis: an event-triggered real-time byzantine-resilient protocol suite. *IEEE Transactions on Parallel and Distributed Systems*, 32(9):2277–2290, 2021.

[18] Antonio Lima, Francisco Rocha, Marcus Völp, and Paulo Esteves-Veríssimo. Towards safe and secure autonomous and cooperative vehicle ecosystems. In *Proceedings of the 2nd ACM Workshop on Cyber-Physical Systems Security and Privacy*, pages 59–70, 2016.

[19] Upstream Security Ltd. Upstream Security's 2022 Global Automotive Cybersecurity Report. Technical report, 2022.

[20] Kirsten Matheus and Thomas Königseder. *Automotive Physical Layer Technologies*, page 134–226. Cambridge University Press, 3 edition, 2021.

[21] Habeeb Olufowobi, Sena Hounsinou, and Gedare Bloom. Controller area network intrusion prevention system leveraging fault recovery. In *Proceedings of the ACM Workshop on Cyber-Physical Systems Security & Privacy*, pages 63–73, 2019.

[22] Johannes Deichmann Ondrej Burkacky and Jan Paul Stein. Automotive software and electronics 2030. 2019.

[23] D. Powell, G. Bonn, D. Seaton, P. Verissimo, and F. Waeselynck. The delta-4 approach to dependability in open distributed computing systems. In *[1988] The Eighteenth International Symposium on Fault-Tolerant Computing. Digest of Papers*, pages 246–251, 1988.

[24] Hans P. Reiser and Rudiger Kapitza. Hypervisor-based efficient proactive recovery. In *2007 26th IEEE International Symposium on Reliable Distributed Systems (SRDS 2007)*, pages 83–92, 2007.

[25] Daniel Ren and Pearl Liu. Fatal crash involving nio's autopilot function sparks online war of words among chinese electric car owners over safety. 2021.

[26] Nick Santhanam Ryan Fletcher, Abhijit Mahindroo and Andreas Tschiesner. The case for an end-to-end automotive software platform. 2020.

[27] Fred B. Schneider. *Replication Management Using the State-Machine Approach*, page 169–197. ACM Press/Addison-Wesley Publishing Co., USA, 1993.

[28] David Shepardson. U.s. safety agency probes 10 tesla crash deaths since 2016. 2021.

[29] Ali Shoker. Exploiting universal redundancy. In *2016 IEEE 15th International Symposium on Network Computing and Applications (NCA)*, pages 199–203, 2016.

[30] Ali Shoker, Jean-Paul Bahsoun, and Maysam Yabandeh. Improving independence of failures in bft. In *2013 IEEE 12th International Symposium on Network Computing and Applications*, pages 227–234. IEEE, 2013.

[31] Ivan Studnia, Vincent Nicomette, Eric Alata, Yves Deswarte, Mohamed Kaâniche, and Youssef Laarouchi. Survey on security threats and protection mechanisms in embedded automotive networks. In *2013 43rd Annual IEEE/IFIP Conference on Dependable Systems and Networks Workshop (DSN-W)*, pages 1–12. IEEE, 2013.

[32] Giuliana Santos Veronese, Miguel Correia, Alysson Neves Bessani, Lau Cheuk Lung, and Paulo Verissimo. Efficient byzantine fault-tolerance. *IEEE Transactions on Computers*, 62(1):16–30, 2011.

[33] Yang Xiao, Ning Zhang, Wenjing Lou, and Y Thomas Hou. A survey of distributed consensus protocols for blockchain networks. *IEEE Communications Surveys & Tutorials*, 22(2):1432–1465, 2020.