Physical Layer Security based on Timing and Voltage Features for Controller Area Networks

A Thesis Submitted for obtaining
the Scientific Title of PhD in Engineering
from
Politehnica University Timișoara
in the Field of
Computer and Information Technology
by

Eng. Lucian-Tudor POPA

PhD Committee Chair:	Prof. Dr. Eng. Mihai Victor MICEA
PhD Supervisor:	Prof. Dr. Eng. Bogdan-Ioan GROZA
Scientific Reviewers:	Prof. Dr. Eng. Ion BICA
	Prof. Dr. Eng. Alin-Dumitru SUCIU
	Prof. Dr. Eng. Daniel-Ioan CURIAC

Date of the PhD Thesis Defense: 20-October-2023

Acknowledgement

This thesis has been developed during my tenure at the Faculty of Automation and Computing at Politehnica University Timișoara, Romania.

Firstly, I would like to express my sincere appreciation to my supervisor, Professor Bogdan Groza, for his unwavering support and guidance throughout my research endeavors. I am grateful to him for involving me in the research projects he coordinated (CSEAMAN and PRESENCE) over the past years and for his invaluable contributions that serve as the bedrock of our research endeavors.

I want to express my profound gratitude to Associate Professor Pal-Stefan Murvay for our fruitful collaboration on various research papers and I also extend my heartfelt thanks to Adriana Berdich, Camil Jichici, Tudor Andreica, and Adrian Musuroi, my colleagues within the research group under Professor Groza's guidance.

Last but not least, I would like to express my deepest gratitude to my family and close friends for their unwavering support throughout my doctoral journey.

Timişoara, October 2023 Lucian-Tudor POPA

Popa, Lucian-Tudor

Physical Layer Security based on Timing and Voltage Features for Controller Area Networks

Keywords:
elliptic curve cryptography, cryptographic key exchange, frame scheduling optimization, time-covert authentication, clock skews, voltage characteristics, physical layer fingerprinting, automotive digital twin, electrical wiring induced noise

Abstract:
Since Controller Area Network (CAN) buses used in vehicles are exposed to certain threats that are described in research works and exploited in real-world conditions, certain updates with emphasis on its security are required. The thesis includes an overview of the CAN vulnerabilities as well as research proposals for their mitigation through physical layer security using timing and voltage characteristics. An innovative key-exchange method that makes use of CAN messages and the CAN protocol requirements for exchanging session keys between nodes is presented as part of the thesis. An improvement for a previous work related to time-covert authentication methods by optimizing the frame transmission times is also discussed in the thesis. From a hardware standpoint, there are certain research papers that use clock skews for periodic CAN messages as fingerprints for transmitter authentication. Other works use the unique voltage characteristics for both periodic and on-event CAN messages as fingerprints for the senders. The thesis includes a broad comparison of the reliability of clock skews and voltage characteristics as fingerprint sources from 9 passenger vehicles. The analysis is done on a public dataset of both frame timestamps used for clock skew derivation and voltage samples used for extraction of unique voltage characteristics for each node. Considering that realistic CAN architectures need to be realized as experimental setups, a digital twin for a real-world vehicle CAN network is described in the later part of the thesis. Considering the noise factor in voltage fingerprinting activities, an analysis of the wiring influence from the digital twin experimental setup with other setups and the real-world vehicle conditions is also presented in the thesis.

Contents

Chapter 1

Introduction

Controller Area Networks (CAN) are still the most commonly used communication channels inside vehicles. Before they were introduced in the 1980s, electronics inside vehicles were communicating using point-to-point connections with separate wires. In contrast, the CAN protocol allows multiple nodes to transmit and receive data using only two wires helping to reduce the cost and complexity of network communication. Another advantage of the CAN bus is its reliability in automotive environments due to the differential wires that define its physical layer. Due to its reliability and cost efficiency, the CAN bus also has more recent updates like CAN-FD (since 2012) and CAN-XL (since 2018), which proves the long-lasting presence of CAN in future vehicles.

1.1 Motivation

CAN is used as a communication medium for both safety and non-safety related systems, but one of its major vulnerabilities is related to communication security since CAN had no such requirements. Considering that passenger vehicles have evolved from mechanical components to electric and electronics that execute multiple software modules inside various operating systems, the security risks have increased as well. There are various proposals regarding the integration of security mechanisms for Controller Area Networks, both from research works and from the automotive industry, which were used as the foundation for the studies presented in this thesis.

Controller Area Network security is the main motivation topic for all the research works that are discussed in the thesis. Since CAN does not provide a mechanism for transmitter identification, genuine nodes that communicate on the CAN bus can be easily impersonated by adversarial nodes that can take control over various vehicle functionalities. Several research works have detailed multiple vulnerabilities concerning the Controller Area Networks which are used as in-vehicle communication buses [1, 2]. In [1], the authors identified gaps in the CAN bus such as missing source identifica-

tion/authentication fields, they performed various packet sniffing, targeted probing and fuzzing attacks on several Electronic Control Units (ECUs), such as the Body Control Module (BCM), Engine Control Module (ECM) or the Electronic Brake Control Module (EBCM). Checkoway et. al. [2] analyzes and evaluates an attack path provided by Tire-Pressure Monitoring Sensors (TPMS) and exploits the Telematics ECU to inject adversarial CAN packets on the internal vehicle networks where this ECU is connected. Considering the identified threats, the AUTOSAR specifications have included the Secure On-Board communication (SecOC) [3] functionality starting with Release 4.2.2 in 2014 [4]. This functionality requires ECUs to include security data in the CAN frames as authentication codes using a freshness counter, also included in the frame, and a secret key that is pre-shared between nodes. The rationale for the SecOC is to protect legitimate communication against injection and replay attacks that have been previously studied. The last decade includes various software-based proposals for securing CAN communication, such as using message authentication [5], [6], [7], identifier reallocation [8], [9], [10], [11], etc. The identifier reallocation method requires periodic changes of frame identifiers in order to protect transmitters of the frames against attacks through an authentication mechanism that updates the frame identifiers with truncated MACs.

CAN bus attacks have many other implications. For example, more recently, several vehicles were stolen using CAN injection attacks [12]. This was possible due to an attack path that is available on the CAN network close to the headlights and near the front bumper of the vehicle. The thief was able to unlock the doors and start the engine using a malicious device that transmits the expected CAN frame sequence. This has later been reported as a known vulnerability and was added to the CVE (common vulnerabilities and exposures) with the unique identifier *CVE-2023-29389* [13]. Other recent weaknesses related to Controller Area Networks from real-world vehicles that were also reported as vulnerabilities are *CVE-2017-14937* [14] and *CVE-2018-9322* [15]. The first vulnerability is caused by predictable security access to the CAN bus that affects airbag units. The second one is reported for the infotainment component of BMW vehicles that can be maliciously used to inject frames on the internal CAN networks.

Another access point for internal CAN buses is the OBD-II/DLC port, which is commonly used for vehicle diagnostics. An OBD-II port from a real-world vehicle is shown in Figure 1.1 that is usually positioned under the dashboard. In many vehicles that are on the road today, this port is not isolated from the rest of the ECUs inside the car. Authors from [16] have presented a comprehensive study that shows the vulnerabilities of 77 OBD-II dongles together with 21 companion applications. Hence, these dongles can be used as an attack vector that injects malicious CAN frames into regular bus traffic. A similar study done by the same authors [17] show how these dongles are used to send malicious commands using CAN frames on two vehicles, a Toyota RAV4 and a Toyota Corolla. Another research group has presented in [18] how diagnostic security keys were extracted from a Fiat Grande Punto by breaking the challenge-response authentication mechanism on the high-speed CAN bus, accessible via the OBD-II connector. Remote

Figure 1.1: OBD-II/DLC port from a real-world vehicle

control of functionalities of a Tesla vehicle, from wireless access to the CAN bus, has been studied by authors in [19]. Fuzzing attacks on the CAN bus targeting an instrument cluster have been presented by authors in [20]. The diagnostic interface can be used to access other internal networks from the vehicle that are connected through gateway ECUs from the OBD-II/DLC network. For example, authors from [21] propose an update of the FlexRay frames to include a lightweight authenticated encryption scheme that will protect the FlexRay communication from OBD-II originated attacks. A summary of attack interfaces for in-vehicle buses is shown by the authors in [22] with related references for CAN and OBD-II and also for Local Interconnect Network (LIN), Media Oriented Systems Transport (MOST) and FlexRay. An injection attack performed via the OBD-II interface is presented by authors in [23]. In Figure 1.2 we show an example of three in-vehicle buses, two CAN buses and a FlexRay network, that communicate over gateways. This network configuration outlines various ECUs like the Accessory Protocol Interface Module, Power Steering Control Module, Instrument Panel Cluster, Restraints Control Module and Anti-lock Brake System from the Diagnostic CAN Bus. The Body CAN bus connects four ECUs, the Body Control Module, Instrument Panel Cluster, Air Conditioning Unit and Rain Light Sensor. The FlexRay network is the communication interface for the Anti-lock Brake System, Powertrain Control Module and Transmission Control Module. The gateway between the CAN buses is the Instrument Panel Cluster while the gateway between the Diagnostic CAN Bus and the FlexRay Network is the Anti-lock

Brake System. Considering the aforementioned attacks, the DLC connector from the Diagnostic CAN Bus is highlighted as an intrusion point for the in-vehicle networks. This can lead to various types of attacks performed through open ports or corrupted gateways on any of the ECUs connected to the internal networks.

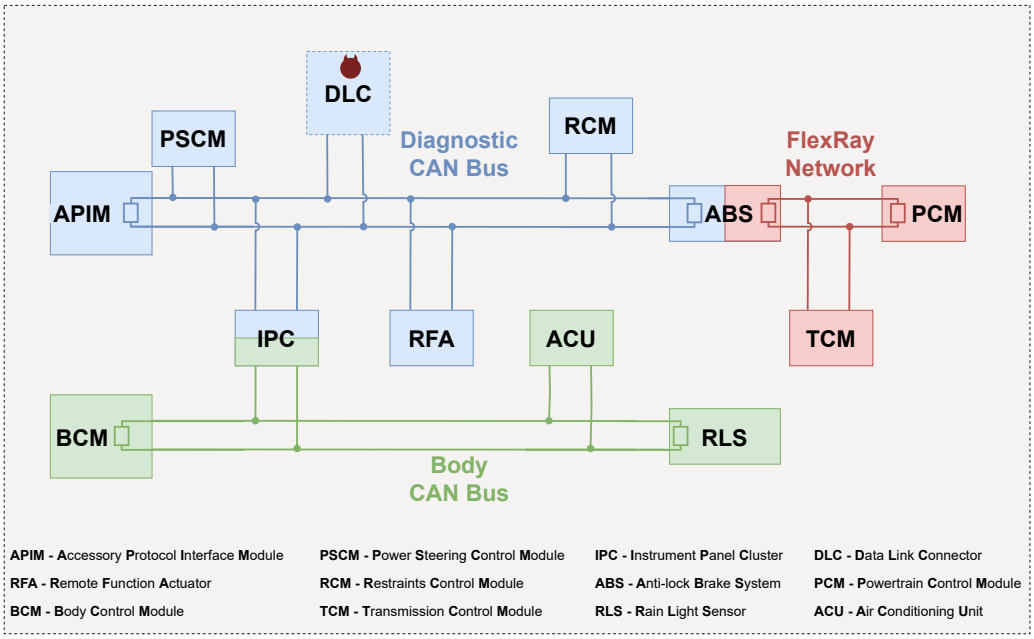


Figure 1.2: Suggestive depiction of an in-vehicle network topology

Such vulnerabilities are the motivation for designing security for Controller Area Networks. One approach that may seem straightforward for securing CAN buses is to implement the data security part in software, transmit it as authentication tags on CAN and verify it using end-to-end checks in software [6]. Unfortunately, CAN frames have a limited size and can include only maximum 8 bytes of payload. Another aspect regarding cryptographic authentication is the use of private and public keys that need to be exchanged, either prior to the communication or during communication. This requires time and bandwidth which may not be available. Another option is to integrate message authentication codes (MACs) inside the CAN frames [3]. The limitation related to the payload size of CAN frames is also relevant for this case. The MACs can be truncated but they require a shared key to be available between nodes. This key can be added to each node during product manufacturing as a pre-shared or it can be dynamically generated each time after vehicle startup as a fresh session key.

All these approaches have their own drawbacks that motivate our search for alternatives, which is the subject of this thesis. A different technique to secure CAN buses is to use covert channels on top of regular communication such as time-covert [24] or voltage-

covert authentication methods [25]. These approaches require nodes to actively authenticate the transmitted frames using secure information that is hidden in time delays or voltage levels. These should be seen as complementary methods to usual CAN communication since they do not require additional signatures or message authentication codes to be part of CAN frames. Nevertheless, authentication of transmitters and frames using voltage-covert channels [25] requires high-performance analog to digital converters. Furthermore, the authentication for time-covert channels is based on sub-microsecond level timers in order to ensure a security level that would ensure CAN communication is secured against adversarial intervention. Using external devices to monitor Controller Area Network communication and extract physical characteristics is another way to authenticate the transmitters [26]. This is based on the characteristics of the transmitters and not necessarily on the communicated data. Even though this method requires additional equipment with high-performance capabilities concerning time capture of CAN frames and voltage sampling of electrical data from the CAN wirings, it is a reliable method that has been shown as efficient by several research works in the last years [26], [27].

1.2 Research objectives

There are several research objectives in this thesis. We start with the key exchange protocols, which are essential in establishing a cryptographic key that can be used to bootstrap security on CAN. We first evaluate elliptic curve cryptography software libraries integrated into automotive controllers, which can be used for key exchange protocols on CAN, and time-covert key-exchanges. Then we pursue time covert authentication and use frame scheduling optimizations to improve the bandwidth of time-covert channels on CAN. Another research area covered in this thesis is related to the physical fingerprinting of Electronic Control Units (ECUs) from real-world vehicles using information collected from the Controller Area Networks, for which we used 9 passenger vehicles having 51 ECUs. The last research topic of the thesis is related to the design and evaluation of a digital twin for a Controller Area Network from a real-world vehicle. The experimental setup that is realized uses automotive-grade embedded boards and wiring harnesses that were retrieved from a real car. The experimental setup is evaluated in the context of wirings required for voltage fingerprinting of nodes in Controller Area Networks. The research objectives of this thesis can be summarized as follows:

1. Literature review on related works for physical security for Controller Area Networks as well as for automotive system digital twins;
2. Evaluation of software libraries with support for elliptic curve cryptography in the context of key-exchange on automotive microcontrollers;
3. Implementation and evaluation of four time-covert key exchange protocols for Controller Area Networks that are fully compatible with existing networks and fully compliant with the standard;

4. Implementation of four frame scheduling optimization algorithms and evaluation of the performance of a time-covert authentication channel in the context of optimized frame transmission;
5. Data collection from four passenger vehicles of frame transmission times for clock skew computation and voltage samples for voltage feature determination in the context of Electronic Control Unit (ECU) fingerprinting;
6. Evaluation of ECU separation from nine passenger vehicles using the determined clock skews and voltage features as well as the environmental impact for these fingerprints;
7. Design and evaluation of an experimental setup that integrates a digital twin for vehicle level functionalities implemented on a Controller Area Network from real-world vehicle cables;
8. Voltage characteristic evaluation of the Controller Area Network from the experimental setup that uses real vehicle cables in comparison with Controller Area Networks from other experimental setups or real-world vehicles in the context of physical fingerprinting.

1.3 Major contributions

This thesis describes several software and hardware methods to implement security for the Controller Area Networks (CAN) used in automotive. The topics that are addressed in the thesis are the elliptic curve evaluation of cryptographic libraries for automotive microcontrollers, key exchange protocols for Controller Area Networks, frame scheduling optimization and time-covert authentication, fingerprinting of ECUs using voltage and timing data from the CAN physical layer and a digital twin design and its evaluation for automotive devices and physical fingerprinting. Considering the objectives defined, the major contributions that this thesis brings are:

1. Timing evaluation of elliptic curve operations by integration of cryptographic libraries on an automotive embedded device [28];
2. Implementation and evaluation of the key-exchange protocols that use CAN frames on automotive grade microcontrollers [29];
3. Implementation and evaluation of the frame scheduling optimization algorithms and time-covert channel authentication protocol on automotive grade microcontrollers [30];
4. Data collection of voltage and clock skew data from the CAN bus from 4 passenger vehicles [31];
5. Analysis of voltage and clock skew fingerprints for 9 passenger vehicles [31];
6. Design and implementation of a Digital Twin for a real-world vehicle CAN network using a wiring harness from a car [32];

7. Evaluation of voltage characteristics of wires used in the various experimental setups and wires from a real-world vehicle [33].

These contributions are part of peer-reviewed publications in conference proceedings and journals. The performance of elliptic curve cryptographic primitives on an automotive microcontroller was evaluated in [28]. Three software libraries were integrated into the source code project for the microcontroller and evaluated with respect to the duration of cryptographic primitives, e.g., key-generation, signature, verification. Two of these software libraries are implemented in C as the programming language while the third one has both C and C++ implementations. It should be pointed out that these software libraries are portable to different development environments. Nevertheless, the time required for the execution of the elliptic curve methods can be considered a shortcoming. To circumvent this, four protocols for exchanging cryptographic keys on the CAN bus based on timing characteristics are designed, implemented and evaluated on an automotive-grade microcontroller in [29]. Two of the protocols rely only on the CAN protocol particularities, i.e., data/remote frames and arbitration, while the other two also depend on the internal hardware timers of the microcontroller. An extension that can be applied to these protocols is presented as an option to increase the security level from the low-entropy exchange key as basis for generating a high-entropy session key. The group version of the proposed protocols and their extension is also briefly described.

Considering the timings for Controller Area Network communication and the size limitation of the data field of its frames, an option to implement security protocols is by using timing information relative to the frames. A time-covert authentication channel was already described in [24] and included in the author's Master Thesis. Unfortunately, the performance of the time-covert channel is reduced by un-optimized traffic due to frame arbitration in case frames are transmitted with a low inter-frame time. An improvement that can be considered for the time-covert channel is through optimizing the frame scheduling on Controller Area Networks as later done by the author in [30]. There are four frame scheduling optimization algorithms that are presented in this work, with details related to optimal values for the minimum and maximum inter-frame times. One of the frame scheduling algorithms is analyzed in the context of adversarial models with both optimized and un-optimized traffic and with single or multiple nodes implementing the protocol. The time-covert channel data rate, security level and the impact of the frame scheduling algorithms on the worst-case arrival times are shown.

Even though time-covert channels are a good method for authenticating frames transmitted on the CAN bus, they still carry small amounts of entropy and their security level is low. Authenticating the frame transmitters is also possible through physical fingerprints that are studied by the authors in [31]. The work presents specific limitations with regards to using physical fingerprints alone, i.e., clock skews or only 1-2 voltage characteristics, and the effects of environmental changes to the initial physical fingerprints. The major contribution of the study is the number of collected and evaluated fingerprints since it is done on 9 passenger vehicles from which 51 ECUs are identified using a dataset of

physical samples from $\sim$400 different frame identifiers. The values determined for the clock skews and voltage features are also presented in a supplemental material in the work as well as in this thesis.

Since vehicle-level functionalities are usually tested on an experimental setup, it is recommended that the setup is as close as possible to the real implementation in the car. In this regard, the work from [32] proposes a Digital Twin for automotive Electronic Control Units (ECUs) that communicate on CAN. The contribution of this work is the design of an experimental setup that contains automotive-grade embedded boards connected on a CAN bus that is part of three wiring harnesses that were removed from a real-world vehicle. The evaluation of the Digital Twin models is done as a statistical analysis by comparing its vehicle speed and engine speed outputs with those from a real-world car when providing the same input for both, i.e., brake input. Possible applications for the CarTwin [32] are suggested in the context of cyber-security and functional safety studies. Furthermore, as the results from [33] show, the experimental setup designed for CarTwin [32] can be used for voltage fingerprinting studies. That is, because, compared to other experimental setups from previous works [34], [35] and to real-world conditions [31], the voltage characteristics of the CAN bus from the CarTwin [32] setup are very close to those from the cars.

The author has contributed to 16 research papers:

1. **L. Popa**, B. Groza, and P.-S. Murvay, "Performance Evaluation of Elliptic Curve Libraries on Automotive-Grade Microcontrollers", in Proceedings of the 14th International Conference on Availability, Reliability and Security, 2019, pp. 1–7,
2. B. Groza, **L. Popa**, and P.-S. Murvay, "TRICKS—Time TRIggered Covert Key Sharing for Controller Area Networks", IEEE Access, vol. 7, pp. 104 294–104 307, 2019,
3. B. Groza, **L. Popa**, and P.-S. Murvay, "CANTO-Covert AutheNtication with Timing channels over Optimized traffic flows for CAN", IEEE Transactions on Information Forensics and Security, vol. 16, pp. 601–616, 2020,
4. **L. Popa**, B. Groza, C. Jichici, and P.-S. Murvay, "ECUPrint—Physical Fingerprinting Electronic Control Units on CAN Buses Inside Cars and SAE J1939 Compliant Vehicles", IEEE Transactions on Information Forensics and Security, vol. 17, pp. 1185–1200, 2022,
5. **L. Popa**, A. Berdich, and B. Groza, "CarTwin—Development of a Digital Twin for a Real-World In-Vehicle CAN Network", Applied Sciences, vol. 13, no. 1, p. 445, 2022,
6. **L. Popa**, C. Jichici, T. Andreica, P.-S. Murvay, and B. Groza, "Impact of Wiring Characteristics on Voltage-based Fingerprinting in Controller Area Networks", May 2023, accepted for publication at IEEE 17th International Symposium on Applied Computational Intelligence and Informatics (SACI 2023),
7. B. Groza, **L. Popa**, and P.-S. Murvay, "INCANTA-INtrusion detection in Controller Area Networks with Time-covert Authentication" in Security and Safety In-

terplay of Intelligent Software Systems: ESORICS 2018 International Workshops, ISSA 2018 and CSITS 2018, Barcelona, Spain, September 6–7, 2018, Revised Selected Papers. Springer, pp. 94–110, 2019,

8. P.-S. Murvay, **L. Popa**, and B. Groza, "Accommodating Time-Triggered Authentication to FlexRay Demands", in Proceedings of the Third Central European Cybersecurity Conference, 2019, pp. 1–6.,
9. B. Groza, **L. Popa**, and P.-S. Murvay, "CarINA-Car sharing with IdeNtity based Access control re-enforced by TPM", in Computer Safety, Reliability, and Security:SAFECOMP 2019 Workshops, ASSURE, DECSoS, SASSUR, STRIVE and WAISE, Turku, Finland, September 10, 2019, Proceedings 38. Springer, pp. 210–222,
10. B. Groza, H. Gurban, **L. Popa**, A. Berdich, and S. Murvay, "Car-to-Smartphone Interactions: Experimental Setup, Risk Analysis and Security Technologies", in 5th International Workshop on Critical Automotive Applications: Robustness & Safety, 2019,
11. B. Groza, **L. Popa**, and P.-S. Murvay, "Highly Efficient Authentication for CAN by Identifier Reallocation With Ordered CMACs", IEEE Transactions on Vehicular Technology, vol. 69, no. 6, pp. 6129–6140, 2020,
12. A. Musuroi, B. Groza, **L. Popa**, and P.-S. Murvay, "Fast and Efficient Group Key Exchange in Controller Area Networks (CAN)", IEEE Transactions on Vehicular Technology, vol. 70, no. 9, pp. 9385–9399, 2021,
13. B. Groza, **L.Popa**, P.-S. Murvay, Y. Elovici, and A. Shabtai, "CANARY-a reactive defense mechanism for Controller Area Networks based on Active RelaYs.", in USENIX Security Symposium, pp. 4259–4276, 2021,
14. P.-S. Murvay, **L. Popa**, and B. Groza, "Securing the Controller Area Network with covert voltage channels", International Journal of Information Security, vol. 20, no. 6, pp. 817–831, 2021,
15. B. Groza, P.-S. Murvay, **L. Popa**, and C. Jichici, "CAN-SQUARE-Decimeter Level Localization of Electronic Control Units on CAN Buses", in Computer Security – ESORICS 2021: 26th European Symposium on Research in Computer Security, Darmstadt, Germany, October 4–8, 2021, Proceedings, Part I 26. Springer, pp. 668–690,
16. B. Groza, **L. Popa**, T. Andreica, P.-S. Murvay, A. Shabtai, and Y. Elovici, "Panopti-CANs - Adversary-Resilient Architectures for Controller Area Networks", in Computer Security–ESORICS 2022: 27th European Symposium on Research in Computer Security, Copenhagen, Denmark, September 26–30, 2022, Proceedings, Part III. Springer, pp. 658–679.

The first 6 of these papers are the main pillars for this thesis while the others also address automotive security topics in the context of Controller Area Networks, FlexRay [36], Trusted Platform Modules and interactions between the vehicle and smartphones. One research paper written shortly before the author has started his PhD studies is [24].

A time-covert authentication channel for CAN frames is presented as part of the study. Its evaluation is performed on an automotive-grade microcontroller by using a free bus, i.e., no additional traffic, and a bus that already has frames replayed from a trace collected from a real car CAN bus. Time-triggered authentication on a different automotive network, FlexRay, has been studied in [37]. The authentication methods are either based on an existing variant [38] or one newly proposed that uses one-way key chains. Both methods require authentication tags to be transmitted as part of the frame or using a separate frame. The evaluation of the authentication methods is done on two automotive-grade embedded platforms. Voltage-covert channels have been proposed in [25] with the novel idea of embedding secure bits on the physical channel of the CAN bus. Localization methods for detecting intrusions on CAN buses by using propagation delays have been proposed and studied in [35] on an experimental setup with automotive-grade embedded platforms and a real-world vehicle.

A different technology that is in use on automotive units is represented by the trusted platform modules (TPMs). A study that relies on TPMs for vehicle sharing using mobile devices with identity based signatures is done in [39] while interaction between mobile devices and vehicles is also studied in [40]. A software-based implementation for dynamically changing the frame identifiers for CAN messages and preserving their priority on the bus is proposed in [8]. The method is designed to use a cipher message authentication code with the Advanced-Encryption Standard (CMAC-AES) based on a pre-shared key between the nodes. A group key exchange algorithm for the CAN bus that uses elliptic curve cryptography, evaluated on two automotive-grade embedded platforms, is presented in [41].

New ideas related to dynamic CAN topology changes are presented in two different works [42, 43]. The first one, [42], presents and evaluates an experimental setup with a CAN network with multiple nodes which are connected with relays in the bus topology. Using the relays, one of the nodes that is a bus guardian, can change the network topology from bus to star, isolating the intruder on one side of the bus or dynamically separating the intruder from one side of the bus to another. The second work [43] improves the design of the experimental setup from the first by reducing the number of wires required and providing new variants for intruder separation or isolation. In this work, a decentralized version is also proposed, with the option of having all nodes switching the topology from bus to daisy-chain.

1.4 Thesis organization

The motivation, objectives and major contributions for the thesis are presented in Chapter 1. A brief background related to CAN is presented in Chapter 2. Then, Chapter 3 begins with the presentation of the timing evaluation for three elliptic curve cryptography software libraries on an automotive embedded platform. In the same chapter, a time-covert

key exchange protocol that is based on four different methods is presented. In Chapter 4, four scheduling optimization algorithms for periodic CAN frames are described while the performance evaluation of a time-covert channel authentication protocol that is implemented together with one of the scheduling optimization algorithms is presented. Chapter 5 presents the physical fingerprinting of ECUs from multiple passenger vehicles that is performed using both voltage features and clock skews, considering the effect of environmental variations on the collected datasets. In Chapter 6, a Digital Twin for several automotive ECUs is described with the use of embedded devices and a real-world vehicle harness. In the same chapter, the voltage characteristics of the CAN bus from the vehicle harness are compared with voltage characteristics from other experimental setups and a real-world vehicle ECU. Chapter 7, holds the conclusions of the thesis.

Chapter 2

Brief Background on Controller Area Networks

Electronic Control Units (ECUs), sensors and actuators from vehicles exchange signal data using CAN (Controller Area Networks). There are two versions of the CAN protocols that are used in vehicles, the low-speed CAN and high-speed CAN. The high-speed CAN is the version that is presented in this section and discussed in all of the thesis chapters.

2.1 CAN physical layer and bit encoding

The bit rate for high-speed CAN is of up to 1Mbps with a physical channel represented by a pair of twisted wires. Nodes that communicate on the CAN bus transmit bits which are either recessive or dominant. The bits represent bus logical states that are either 1 or 0 and are grouped into specific frame structures that start with the frame header followed by data, checksum and acknowledge bit fields. The nodes use integrated circuits such as microcontrollers to exchange data using the CAN transmission (CAN-TX) and reception (CAN-RX) lines which are usually of $5V$ and $0V$ as voltage levels for the recessive and dominant bits. The CAN-TX and CAN-RX lines are connected to a CAN transceiver which links the node to the physical medium. On the wires, called CAN-High (CAN-H) and CAN-Low (CAN-L), the voltage level is around $2.5V$ while the bus is idle or when recessive bits are transmitted. The voltage level increases to $\sim 3.5V$ on CAN-H and decreases to $\sim 1.5V$ on CAN-L whenever a dominant bit is transmitted. The bit states and corresponding voltage levels for CAN-H and CAN-L on a CAN bus with the bit rate of 500Kbps are shown in Figure 2.1.

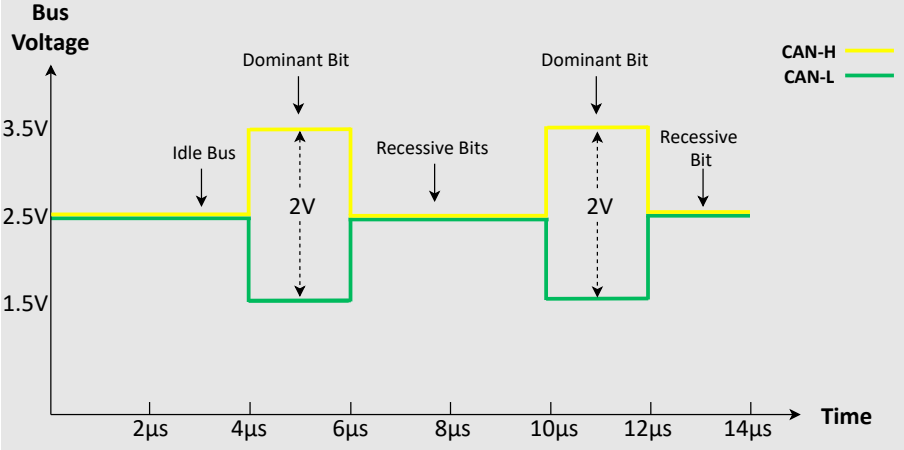


Figure 2.1: Bit states and voltage levels for CAN-H and CAN-L on a CAN bus with the bit time of $2\mu s$

2.2 CAN data and remote frames

As previously stated, the CAN messages have a specific structure. There are four types of messages defined for CAN communication. These are the data frames, remote frames, error frames and overload frames. For data and remote frames, the CAN header begins with the start of the frame bit (SOF) and an arbitration field with the frame identifier. It is denoted as an arbitration field because, during the arbitration phase described in the CAN standard [44], multiple nodes are allowed to transmit bits on the CAN bus at the same time. Due to the hardware design of the CAN transceivers, recessive bits are overwritten by dominant bits. This means that frames with a lower value of the frame identifier "win" the arbitration and continue the frame transmission. The frame header ends with the control field that specifies the number of bytes that are sent in the data field. Depending on the value from the control field, 0 to 8 bytes can be transmitted inside a CAN frame as payload. The data field is followed by the CRC (cyclic redundancy check) bit field that contains a 15-bit CRC which is used to verify the integrity of all bits transmitted in the frame header and data area. After the CRC part, there is an acknowledge field that is left as recessive by the transmitter and set to dominant by all receivers that have checked the frame integrity and successfully received the frame information. CAN frames end with an EOF (end of frame) field of 7 recessive bits. The bit fields for CAN2.0 data frames with standard (11-bit) and extended (29-bit) identifiers are shown in Figure 2.2. The bit fields from the frame header differ for standard and extended identifiers. The arbitration field contains the IDE (identifier extension) bit for extended frames while the control field contains the IDE for standard frames. The arbitration field for extended frames contains an additional bit, the SRR (substitute remote request), that is not present

in the arbitration field for standard frames. The RTR bit is the remote transmit request bit which is dominant for data frames and recessive for remote frames. The bits r0 and r1 are reserved bits, and require a dominant state during CAN communication.

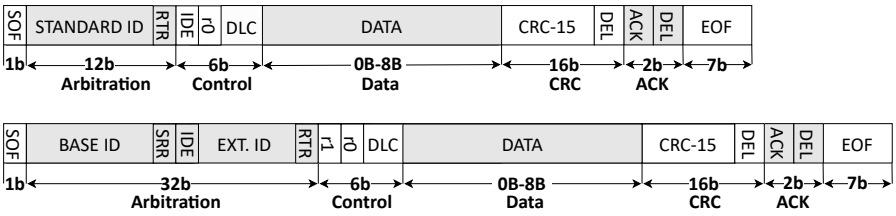


Figure 2.2: Frame structure and bit fields illustrated for standard CAN data frames (top) and extended CAN data frames (bottom)

2.3 CAN error frames and error states

There are specific rules regarding the protocol described in the CAN standard [44] that need to be applied by nodes that communicate on CAN. The errors that may be noticed during CAN communication are:

- Bit error, when the node that transmits a data/remote frame on the bus reads a different bit state from the physical layer than the one transmitted,
- Stuff error, when the node that transmits data on the bus violates the stuffing bit rule by sending more than five consecutive bits with the same polarity, i.e., recessive or dominant,
- Form error, when the node that transmits data on the bus violates the form rule by sending a different bit state than expected for specific bits that are reserved, delimiter bits, end of frame bits, etc.,
- CRC error, when the information transmitted in the CRC bit field is different from the one computed by the nodes receiving the data,
- ACK error, when the transmitter reads back the acknowledge bit as recessive, i.e., there is no active receiver of the frame.

Any violation of the rules specified in the CAN standard [44] will result in an error frame reported by the frame transmitter (Bit error, ACK error) or its receivers (Stuff error, Form error, CRC error). An error frame may have between 6 and 12 bits in the error flag area followed by 8 bits which represent the error delimiter. The error flag and error echo bit fields may contain dominant or recessive bits, depending on the type of error flag transmitted. Error frames which contain dominant bits in these bit fields are called active error frames while error frames which contain recessive bits in these bit fields are called

passive error frames. The type of error frame transmitted by the nodes is related to their error state. The error delimiter contains only recessive bits. All error frames are reported during a data or remote frame transmission. The bit fields for an error frame are shown in Figure 2.3.

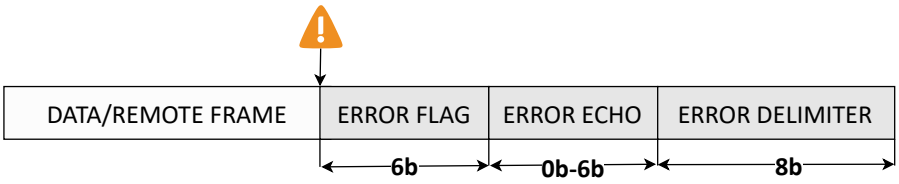


Figure 2.3: Frame structure and bit fields illustrated for error frames reported after protocol violation during data or remote frame transmission

There are three error states defined in the CAN standard [44] that depend on two counters which are used by each node during active communication. These counters are denoted as transmission error count (TEC) and receive error count (REC) and are incremented whenever error frames are reported by nodes for any violation of the CAN protocol during data or remote frames transmission. For an error frame reported during transmission, the transmitter increments its TEC with 8 while receivers of that frame increment their REC with 1. The error states which are shown in Figure 2.4 are defined as:

- Error active, when both TEC and REC values are smaller than 127, the first state after startup where nodes are allowed to transmit active error frames,
- Error passive, when either TEC or REC is higher than 127, state in which nodes are allowed to transmit passive error frames,
- Bus off, when TEC value is higher than 255, state in which nodes cannot transmit error frames and are disconnected from bus communication.

2.4 Stuffing bits during CAN communication

Since the CAN physical layer only has the CAN-H and CAN-L lines with no clock line for data transmission, it is asynchronous. Thereby, nodes that are active on the CAN bus are required to re-synchronize during communication. The re-synchronization is done during bit state changes from recessive to dominant. Since consecutive bits may be transmitted with the same polarity, either dominant or recessive, bit stuffing is required according to the CAN standard [44]. This method requires transmitters to add a bit with a different polarity after five bits with the same polarity. If there are five consecutive dominant bits transmitted, a recessive stuff bit is added. The same logic applies to consecutive

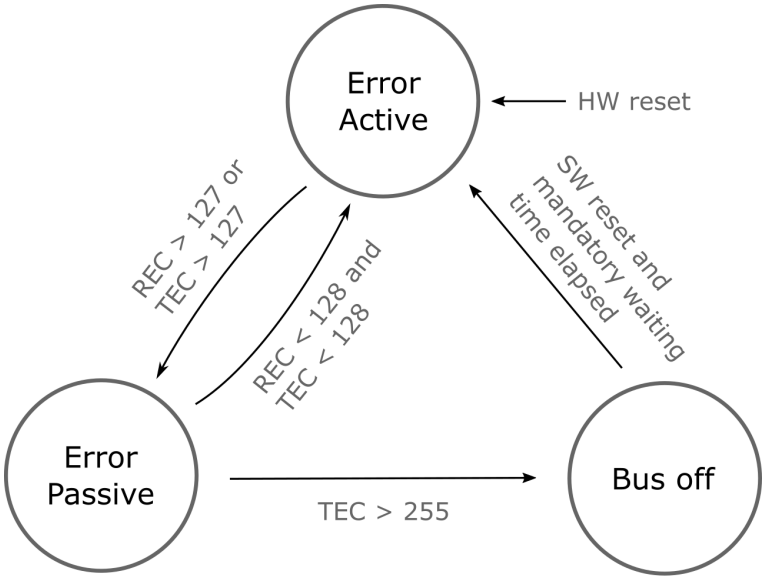


Figure 2.4: Error states defined for CAN nodes and conditions to change the error state

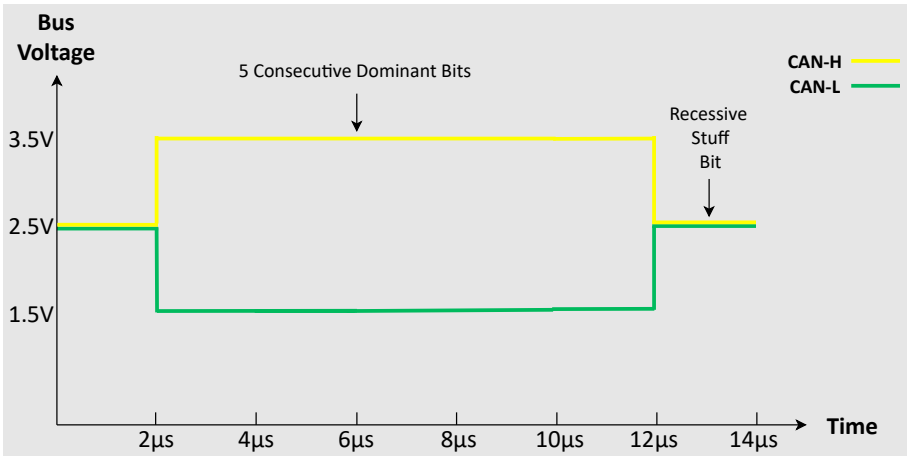


Figure 2.5: Bit stuffing performed by the transmitter after five consecutive dominant bits on a CAN bus with the bit time of $2\mu s$

recessive bits after which a dominant stuff bit is added. Receivers are required to perform de-stuffing so the stuff bits are not considered as part of the frame content and are only required on the physical layer to allow active node re-synchronization. An example of the bit stuffing mechanism is shown in Figure 2.5 where a recessive bit is transmitted for

stuffing after five consecutive dominant bits.

2.5 Other aspects regarding CAN communication

For the CAN communication channel, there are at least two nodes required to be active. This is due to the acknowledgement of frames which takes place only if there is an active receiver for the transmitter of a CAN frame. Otherwise, the transmitter will continuously report error frames due to the missing acknowledgement. Whenever multiple nodes transmit bits during the arbitration field, only one will continue to send the entire frame. All other nodes have to wait for frame transmission to end and try to pass the arbitration field and transmit their frame. If a data frame and a remote frame with the same identifier are transmitted at the same time on the bus, the data frame will be the only one transmitted on the bus because the node which transmits a remote frame will not try to perform re-transmission since it received the data frame during the request time.

Chapter 3

Time-Covert Key Exchange on the CAN Bus

This chapter is based on two previous research papers of the author, [28] and [29]. The first research paper [28] presents the evaluation of key exchange algorithms based on elliptic curve cryptography (ECC) by using open source libraries on an automotive embedded platform. This paper aims to demonstrate the computational costs for such procedures, which may cause significant concerns on embedded platforms. The second paper presents a new method for performing an asymmetric key exchange in order to secure communication on Controller Area Networks [29]. The scope of this paper is to show that key exchange can also be performed based on timing characteristics that can be merged with public-key cryptography to reinforce security or be used in their absence.

3.1 The cost of key-exchange mechanisms based on ECC

In this section, the background and related works for elliptic curve cryptography evaluation are presented. After the background and related works are discussed, the experimental setup is presented and the software libraries that are integrated are detailed. Following the experimental results, the run-time characteristics for the elliptic curve cryptographic primitives are discussed.

3.1.1 Related works

Considering recent papers findings with respect to threats and vulnerabilities for in-vehicle networks [2], [45], an alternative to protect sensitive data exchanged on them is the use of cryptographic protocols. Since the classic AUTOSAR specification already incorporates cryptographic primitives and protocols within the Crypto library (Crypto)

[46] while they are executed as part of the Crypto Service Manager (CSM) [47], the automotive industry is moving forward towards securing vehicle network communication. One way to ensure secure communication on the CAN bus is the implementation of cryptographic protocols that rely on elliptic curves for key exchange and signatures that allow verification using public keys.

Research works that evaluate open-source or custom elliptic curve libraries are presented in what follows. Computational time for MIRACL [48] and RELIC [49] was measured by Pigatto et. al [50] using the Ubuntu operating system that runs on a 2.10GHz Pentium Dual-Code processor. They used key sizes of up to 256 bits and packets of 50KB and 100KB as settings for the evaluated libraries. An analysis that is closer to the work presented in this chapter [28] is done by Ruan de Clercq et. al [51] which evaluate the run-time performance on an ARM Cortex-M0+ platform of a cryptographic library they implement and RELIC [49] using the Koblitz NIST K-233 elliptic curve. Power consumption analysis on a MSP430 platform is performed by Hinterwälder et. al [52] for their implementation of elliptic curve Diffie-Hellmann key-exchange (ECDH) [53] using the EC25519 elliptic curve. Other research works evaluate cryptographic primitives for automotive platforms or for specific automotive use-cases. Thereby, authors from [54] implement an AUTOSAR compliant cryptographic library and evaluate the execution speed on various automotive grade microcontroller platforms. Computational performance for Transport Layer Security (TLS) using elliptic curve digital signature algorithm (ECDSA) [55] is evaluated by Zelle et. al [56] on two Infineon AURIX TC297 platforms that exchange message frames using Automotive Ethernet. The authors utilized wolfSSL [57] as their cryptographic library of choice for conducting their experiments, with the goal of determining the latency introduced by cryptographic operations for data block transfers.

Table 3.1: Evaluated cryptographic protocols from open-source libraries

Library	ECDSA [55]	ECDH Key-Exchange [53]
MIRACL [48]	-	✓
RELIC [49]	✓	✓
wolfSSL [57]	✓	✓

These algorithms can be used for key exchange between nodes that communicate on in-vehicle networks and even for wireless vehicle access to the car using mobile devices. The open-source libraries are integrated in the software build environment from Infineon for automotive-grade AURIX microcontrollers. We evaluate the run-time performance for the primary operations required for elliptic curve digital signature algorithm (ECDSA) [55] and elliptic curve Diffie-Hellmann key-exchange protocol (ECDH) [53]. In this regard, three open source libraries that are available online are analyzed, i.e., MIRACL [48], RELIC [49] and wolfSSL [57] which have support for cryptographic primitives and protocols based on various standardized elliptic curves. Both MIRACL [48] and RELIC [49] software libraries include the implementation for Diffie-Hellmann key-exchange using elliptic curves (ECDH) [53]. The RELIC software library [49] also

includes the implementation of the elliptic curve digital signature algorithm (ECDSA) [55]. Finally, WolfSSL [57] supports both the elliptic curve digital signature algorithm [55] and the elliptic curve Diffie-Hellmann key exchange protocol [53]. The cryptographic operations that were evaluated from the open-source libraries are presented in Table 3.1.

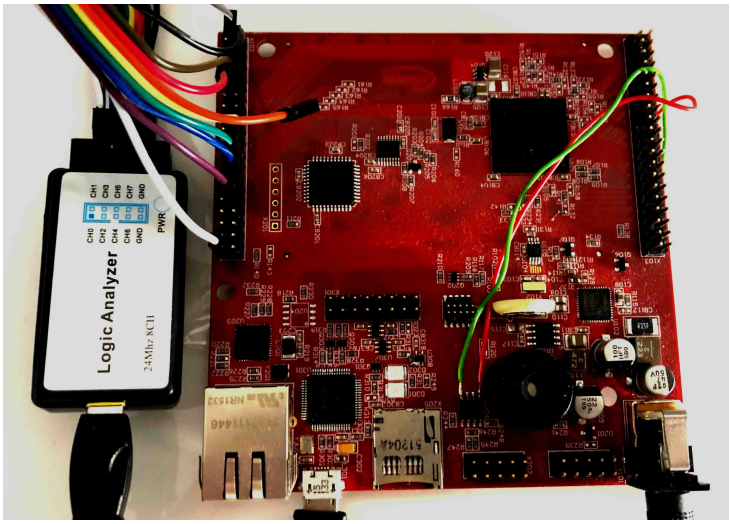


Figure 3.1: Experimental setup for run-time evaluation of elliptic curve algorithms

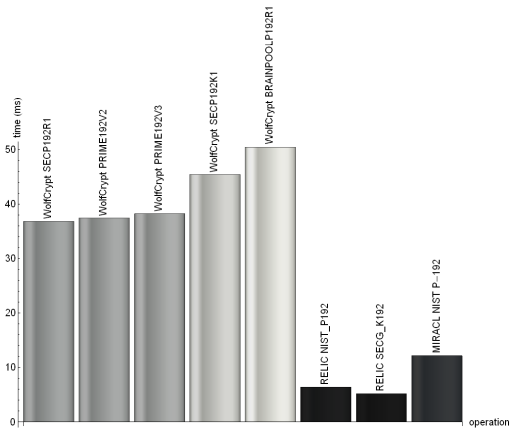


Figure 3.2: Share generation time for ECDH

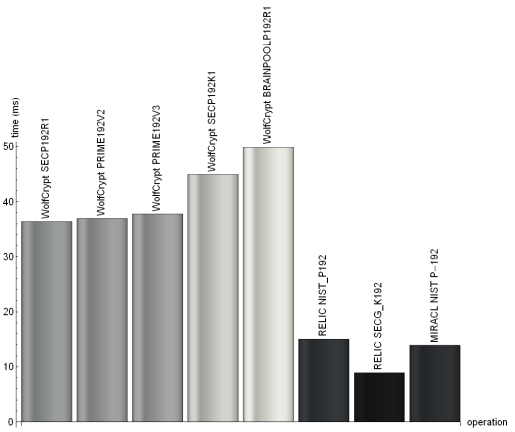


Figure 3.3: Key recovery time for ECDH

3.1.2 Embedded platform setup and integration of software libraries

In this subsection, the embedded platform setup together with the integration details for each software library are described. The evaluation method used for measuring the execution time of each cryptographic primitive is presented after the integration details.

Embedded Platform Description. The Infineon TC297 AURIX application kit shown in Figure 3.1 was used for the experiments, since TC297 is a high-end automotive grade microcontroller used in highway assist applications such as camera vision or radar systems according to its datasheet. This controller integrates a processor that runs three independent cores and 728KB of RAM (Random Access Memory), 384KB of EEPROM (Electrically Erasable Programmable Read-Only Memory) and 8MB of FLASH memory. It also embeds a hardware security module (HSM) as part of the chip that allows cryptographic operations to be executed with a higher speed and the exchange of results to be done by a dedicated CPU with protected memory. The experimental setup used for the evaluation of the elliptic curve cryptographic algorithm run-time is shown in Figure 3.3.

Software Library Integration. The integration of MIRACL [48], RELIC [49] and wolfSSL [57] on the embedded platform is done considering their support for various elliptic curves that are detailed in what follows. The MIRACL [48] includes both C and C++ implementations for elliptic curve cryptographic primitives and libraries. For the elliptic curve Diffie-Hellmann key-exchange protocol (ECDH) [53] implementation, the 192 bit prime field (NIST P-192 elliptic curve) is used

The RELIC [49] cryptographic library has C implementation support for security algorithms like Rivest–Shamir–Adleman (RSA) [58] signatures and encryption, elliptic curve Diffie-Hellmann key-exchange protocol (ECDH) [53] and elliptic curve digital signature algorithm (ECDSA) [55]. The digit size is configured as 32 bit for the TC297 architecture, so all the cryptographic operations work with 32-bit size digits. There are various configurations for the elliptic curves used since the software library supports them. Based on the algorithm that is evaluated, the configuration is done as follows:

- For ECDH key exchange the prime fields used for the elliptic curves have the following bit size (bits): 158, 160, 192, 221, 224, 226, 251, 254, 255, 256, 381, 382, 383, 384, 455, 477, 508, 511, 521, 638, 1536.
- For ECDSA signatures the prime fields used for the elliptic curves have the following bit size (bits): 160, 192, 224, 256, 384.

The wolfSSL [57] software library includes the implementation of SSL/TLS cryptographic protocols in C language. Since wolfCrypt [59] provides the implementation for the cryptographic primitives and algorithms used by wolfSSL [57], such as elliptic curve Diffie-Hellmann key-exchange protocol (ECDH) [53] and elliptic curve digital signature algorithm (ECDSA) [55], the discussion that follows is about wolfCrypt [59] instead of wolfSSL [57]. Similar to the configuration of RELIC, 32-bit size digits were used in wolfCrypt [59]. The elliptic curves used are SECP, PRIME, KOBLITZ and BRAIN-

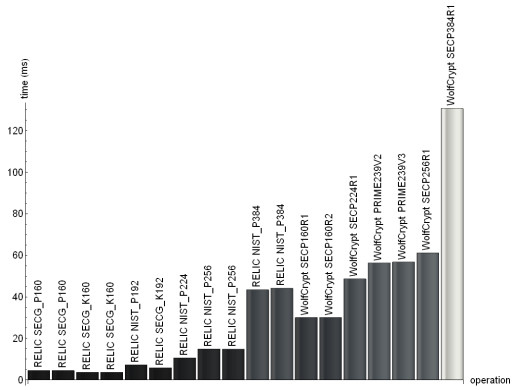


Figure 3.4: Key generation time for ECDSA

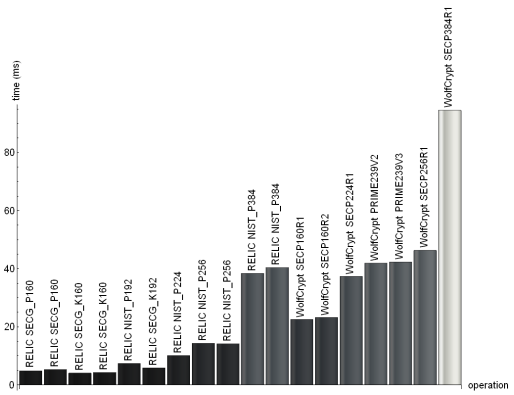


Figure 3.5: Signing time for ECDSA

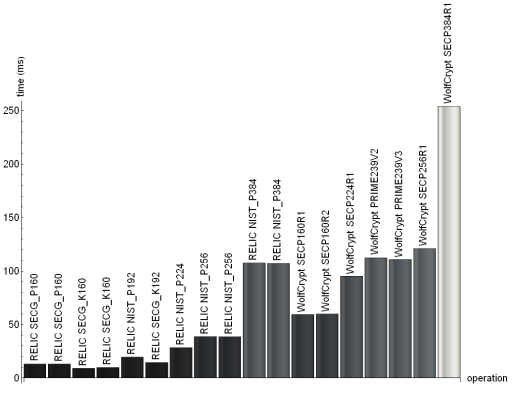


Figure 3.6: Signature verification time for ECDSA

Table 3.2: Steps from cryptographic protocols that are evaluated for each algorithm

Security protocol	Key generation	Signature	Verification	Shared secret
ECDSA [55]	✓	✓	✓	N/A
ECDH [53]	✓	N/A	N/A	✓

POOL for the ECDH and ECDSA operations over prime fields with various bit sizes. These are the 112 bit prime field (SECP elliptic curve), 128 bit prime field (SECP elliptic curve), 160 bit prime field (SECP, KOBLITZ, BRAINPOOL elliptic curves), 192 bit prime field (SECP, PRIME, KOBLITZ, BRAINPOOL elliptic curves), 224 bit prime field (SECP, KOBLITZ, BRAINPOOL elliptic curves), 239 bit prime field (PRIME elliptic curve), 256 bit prime field (PRIME, KOBLITZ, BRAINPOOL elliptic curves), 320 bit prime field (BRAINPOOL elliptic curve), 384 bit prime field (SECP, BRAINPOOL elliptic curves), 512 bit prime field (BRAINPOOL elliptic curve) and 521 bit prime field (SECP elliptic curve).

Considering the support for elliptic curve operations from MIRACL [48], RELIC [49] and wolfCrypt [59] the evaluation of the run-time execution for the following security algorithms using the embedded platform is presented in what follows:

- Elliptic curve Diffie-Hellmann key-exchange [53] protocol for all software libraries,
- Elliptic curve digital signature algorithm [55] for RELIC [49] and wolfCrypt [59].

Embedded Platform Run-Time Evaluation Method. A logic analyzer tool is used for measuring the execution time for each cryptographic function on the Infineon TC297 hardware platform. One probe of the hardware tool is connected to a GPIO pin from the microcontroller exposed on a connector from the development kit. Once the execution of one step from the security algorithm is started, the pin is toggled and at the end of that step, the pin is toggled again. For the elliptic curve Diffie-Hellmann key-exchange protocol [53], the time for key generation and computation of the shared secret between two parties are measured. The time for several steps is measured for the elliptic curve digital signature algorithm [55]. The first step is the key-pair generation, followed by the signature using the private key and, in the end, the verification of the same signature using the public key. The operations that are measured are summarized in Table 3.2.

3.1.3 Experimental results of the embedded platform evaluation

For the elliptic curve Diffie-Hellmann key-exchange protocol (ECDH) [53], evaluation is performed on all elliptic curves supported by MIRACL [48], RELIC [49] and wolfCrypt [59]. In this way, 52 elliptic curves are evaluated, 1 from MIRACL [48], 25 from RELIC [49] and 26 from wolfCrypt [59]. The bit size of the prime fields supported by wolfCrypt [59] is of 112 to 512 bits, while for RELIC [49] it is of 158 bits to 1536 bits and for MIRACL [48] of 192 bits. The evaluation results for the key generation and shared secret computation run-times for all elliptic curves are shown in Table 3.3 (up to 256 bit output size)with a graphic overview presented in Figures 3.2 and 3.3 based on the data for some of the evaluated elliptic curves. Comparing the time required for key generation or shared

secret computation for the NISTP192 elliptic curve, it is noticeable that RELIC [49] implementation performs faster than MIRACL [48] and wolfCrypt [59]. The execution times for key generation are of $6.36ms$ for RELIC [49], $12.12ms$ for MIRACL [48] and $36.8ms$ for wolfCrypt. The execution time for the shared secret computation is $15.06ms$ for RELIC [49], $13.9ms$ for MIRACL [48] and $36.4ms$ for wolfCrypt [59]. Considering other elliptic curves such as SECP224K1 and SECGK224, both of 224 bits, the execution time is also faster for RELIC [49] with $7.71ms$ and $13.01ms$ for the operations compared to wolfCrypt [59] which takes $57.3ms$ and $56.8ms$ for performing the same steps. For a prime field size of 256 bits, the execution times are still better for RELIC [49] with $13.04ms$ and $29.48ms$ for the operations using the NISTp256 elliptic curve compared to wolfCrypt [59] with $60.3ms$ and $59.8ms$ for the same operations performed on the SECP256R1 elliptic curve.

For the elliptic curve digital signature algorithm [55] evaluation, the C implementation from RELIC [49] and wolfCrypt [59] were integrated in the software project for the AURIX microcontroller. A 20 byte message was used as input for computing and verifying the ECDSA signature for all the experiments. There are 14 elliptic curves evaluated, 7 from RELIC [49] and 7 from wolfCrypt [59]. All run-times for ECDSA key generation, signature and verification of signature are shown in Table 3.4. The RELIC [49] library allows selection of different hash functions with the same bit size, i.e., SHA-1 or BLAKE2S-160 with 160 bits and SHA-256 and BLAKE2S-256 with 256 bits (BLAKE is a faster and more compact hash function [60]). A graphical overview of execution times for key generation, signature for a given message and verification of the signature are shown in Figure 3.4, Figure 3.5 and Figure 3.6. Run-time execution of RELIC [49] implementation is lower than that for the execution of the wolfCrypt [59] implementation. For the same prime field size of 224 bits, the operations were executed in $48.7ms$, $37.4ms$, $95.2ms$ for SEC224R1 on wolfCrypt [59] while for NISTP224 on RELIC [49] they were executed in $10.6ms$, $10.1ms$ and $28.31ms$. For the same elliptic curve on RELIC [49], but with different hash functions, the differences are negligible at the order of a few milliseconds.

Based on the evaluation data for ECDH [53] and ECDSA [55] for all software libraries, RELIC [49] has more configuration options and supports more elliptic curves compared to MIRACL [48] and wolfCrypt [59]. Its implementation of elliptic curve cryptographic operations is the one that has faster execution, so it is the most convenient candidate with regards to applications that have time-critical constraints. If there are regulations that require software libraries with certifications, wolfCrypt [59], which is part of wolfSSL [57], can be an option considering that it is FIPS 140-2 certified [61] according to the information on their website. Considering that an application requires a software library with C++ support that implements elliptic curve operations, MIRACL [48] is the option that can fulfill this requirement. In the context of secure key-exchange on Controller Area Networks, one of the limitations is the maximum payload for a CAN frame of 64 bits. This means that, even for the fastest operation for a cryptographic key-

Table 3.3: Operation time for ECDH ordered by output size of up to 256 bits

Library	Elliptic curve	Output size	Operation type	Duration [ms]
wolfCrypt	SECP112R1	112 bits	generate	19.8
			compute shared secret	19.6
wolfCrypt	SECP112R2	112 bits	generate	23.7
			compute shared secret	23.8
wolfCrypt	SECP128R1	128 bits	generate	21.9
			compute shared secret	21.6
wolfCrypt	SECP128R2	128 bits	generate	27.6
			compute shared secret	27.2
RELIC	BNP158	158 bits	generate	3.29
			compute shared secret	6.01
wolfCrypt	SECP160R1	160 bits	generate	30
			compute shared secret	29.6
wolfCrypt	SECP160R2	160 bits	generate	30.2
			compute shared secret	29.8
wolfCrypt	SECP160K1	160 bits	generate	34.6
			compute shared secret	34.3
wolfCrypt	BRAINPOOLP160R1	160 bits	generate	37.5
			shared secret	37.2
RELIC	SECGP160	160 bits	generate	4.01
			compute shared secret	9.78
RELIC	SECGK160	160 bits	generate	3.27
			compute shared secret	5.78
wolfCrypt	SECP192R1	192 bits	generate	36.8
			compute shared secret	36.4
wolfCrypt	PRIME192V2	192 bits	generate	37.4
			compute shared secret	37
wolfCrypt	PRIME192V3	192 bits	generate	38.2
			compute shared secret	37.8
wolfCrypt	BRAINPOOLP192R1	192 bits	generate	50.4
			compute shared secret	49.9
wolfCrypt	SECP192K1	192 bits	generate	45.4
			compute shared secret	45
RELIC	SECGK192	192 bits	generate	5.15
			compute shared secret	8.94
RELIC	NISTP192	192 bits	generate	6.36
			compute shared secret	15.06
MIRACL	NISTP192	192 bits	generate	12.127
			compute shared secret	13.93
RELIC	CURVE22103	221 bits	generate	10.14
			compute shared secret	25.92
wolfCrypt	SECP224R1	224 bits	generate	48.8
			compute shared secret	48.4
wolfCrypt	SECP224K1	224 bits	generate	57.3
			compute shared secret	56.8
wolfCrypt	BRAINPOOLP224R1	224 bits	generate	62.8
			compute shared secret	62.4
RELIC	NISTP224	224 bits	generate	9.28
			compute shared secret	21.31
RELIC	SECGK224	224 bits	generate	7.71
			compute shared secret	13.01
RELIC	CURVE4417	226 bits	generate	12.73
			compute shared secret	31.84
wolfCrypt	PRIME239V1	239 bits	generate	56.6
			compute shared secret	56.2
wolfCrypt	PRIME239V2	239 bits	generate	55.7
			compute shared secret	55.2
wolfCrypt	PRIME239V3	239 bits	generate	56.1
			compute shared secret	55.7
RELIC	CURVE1174	251 bits	generate	13.88
			compute shared secret	35.27
RELIC	BNP254	254 bits	generate	10.52
			compute shared secret	17.54
RELIC	CURVE25519	255 bits	generate	14.72
			compute shared secret	36.7
wolfCrypt	BRAINPOOLP256R1	256 bits	generate	78.4
			compute shared secret	77.9
wolfCrypt	SECP256R1	256 bits	generate	60.3
			compute shared secret	59.8
wolfCrypt	SECP256K1	256 bits	generate	69.1
			compute shared secret	68.7
RELIC	NISTP256	256 bits	generate	13.04
			compute shared secret	29.48
RELIC	BSIP256	256 bits	generate	14.64
			compute shared secret	35.64
RELIC	SECGK256	256 bits	generate	10.36
			compute shared secret	18.07
RELIC	BNP256	256 bits	generate	10.52
			compute shared secret	18.22

exchange, i.e., with a security level of 158 bits, that takes roughly $\sim 10ms$ for generation and computation of the shared secret, CAN frames are required to be exchanged between nodes. This will require additional time for the key exchange between nodes to be established. A different proposal that utilizes CAN frames without payload to exchange keys is described in the following section – this proposal can be used with or without the help of cryptography according to the specific constrains of the platform.

Table 3.4: Operation time for ECDSA ordered by output size of up to 512 bits

Library	Elliptic curve	Hash function	Output size	Operation type	Duration [ms]
wolfCrypt	SECP160R1	SHA-1	320 bits	generate	30
				sign	22.56
				verify	59.3
wolfCrypt	SECP160R2	SHA-1	320 bits	generate	30.1
				sign	23.2
				verify	59.8
RELIC	SECGP160	BLAKE2S-160	320 bits	generate	4.54
				sign	4.86
				verify	13
RELIC	SECGP160	SHA-1	320 bits	generate	4.54
				sign	5.27
				verify	13
RELIC	SECGK160	BLAKE2S-160	320 bits	generate	3.65
				sign	4.01
				verify	8.94
RELIC	SECGK160	SHA-1	320 bits	generate	3.65
				sign	4.26
				verify	9.78
RELIC	NISTP192	SHA-1	384 bits	generate	7.23
				sign	7.34
				verify	19.43
RELIC	SECGK192	SHA-1	384 bits	generate	5.81
				sign	5.83
				verify	14.37
wolfCrypt	SECP224R1	SHA-1	448 bits	generate	48.7
				sign	37.4
				verify	95.2
RELIC	NISTP224	SHA-1	448 bits	generate	10.6
				sign	10.1
				verify	28.31
wolfCrypt	PRIME239V2	SHA-1	478 bits	generate	56.3
				sign	41.9
				verify	112.4
wolfCrypt	PRIME239V3	SHA-1	478 bits	generate	56.8
				sign	42.3
				verify	110.7
wolfCrypt	SECP256R1	SHA-256	512 bits	generate	61.2
				sign	46.3
				verify	121
RELIC	NISTP256	BLAKE2S-256	512 bits	generate	14.9
				sign	14.3
				verify	38.6
RELIC	NISTP256	SHA-256	512 bits	generate	14.8
				sign	14.1
				verify	38.4

3.2 A time-covert key-exchange protocol on the CAN Bus

This section introduces the time-covert key-exchange protocols to which the author contributed in [29]. As stated, this kind of approach can alleviate the need for expensive public-key operations or reinforce such security mechanisms.

3.2.1 Related works

Controller Area Network (CAN) security improvements have been proposed in the past, as also shown in [62]. Some of the existing works propose the inclusion of message authentication codes inside of the frames [63], [64], while others focus on group key authentication methods [6], physical characteristics for authenticating the sender [26], [65] or timing-based security [66], [67]. Whenever message authentication codes are used [64], [63], [68], a shared secret needs to be known by all the parties that verify the authenticity of the message. Except for the research works done by authors in [69] and [70], there was little focus on the key exchange capabilities of the CAN communication channel.

The proposal from Mueller and Lothspeich [69] is the utilization of CAN protocol bit states as a medium to exchange a session key by transmitting random values of 0 and

1 simultaneously on the vehicle bus. The idea behind their proposal is to rely on the physical overwriting of a recessive bit by a dominant bit due to the logic-AND property of the CAN network layer. This means that any combination of bits that are (dominant, recessive), (recessive, dominant) and (dominant, dominant) would result in a dominant state on the bus, so an external party cannot know which node had set the bit state on the bus. For the nodes that communicate, if they transmit a recessive bit and read a dominant state from the bus, they can extract one bit from the random session key. Afterwards, both nodes exchange the complementary value of the generated random bits. In this way, a node that transmits a dominant bit in the first iteration would transmit a recessive bit and will be able to extract the bit state from the other node. This means that a pair of (dominant, recessive) or (recessive, dominant) will remain unknown to an adversary that will not be aware of the bit state transmitted by the nodes on the bus. The genuine nodes are the only ones able to reconstruct the bits read from the bus state as a session key.

The CAN bus requires at least two nodes to be active during communication. A general overview of a method for one of the proposed key-exchange protocols that uses the arbitration mechanism on the CAN physical layer is shown in Figure 3.7. The value of Δ represents a specific time interval when both nodes transmit a frame identifier value that is either smaller ($\mathsf{ID}_{\mathsf{min}}$) or bigger ($\mathsf{ID}_{\mathsf{max}}$) for that iteration. A VN hardware device that is used to log the network traffic is connected to the CAN bus. The shared key between the nodes is also considered, possibly a common practice in the automotive industry, which is denoted as $\widetilde{w}$ in the same figure. An example would be an implementation of a security protocol similar to the secure transient association proposed by authors in [71] that require ECUs inside a vehicle to store a shared secret during the first seconds while the vehicle is started for the first time in the production plant of the car manufacturer. A weak shared secret, from cryptographic security standpoint, is considered as good enough for the proposed methods. For nodes that implement the classic AUTOSAR specifications [46], the shared secret can be strengthened, whenever required, i.e., for Secure On-Board Communication (SecOC) [72]. The protocols that are proposed in what follows allow the exchange of a session key in the application layer that can be further used to re-enforce a weak key and generate a stronger key, as later shown. An extension of the protocols is considered with additional cryptographic operations that will increase the efficiency and security level for the established secure channel. There are four versions for the key exchange that rely on data and remote frames, arbitration and timing-based transmission. For the protocols that rely on timing-based transmission, the possibility to piggy-back the frames with key parts exchange using Diffie-Hellman (DH) version of the Encrypted-Key-Exchange protocol (EKE) [73] or the Simple Password Exponential Key Exchange (SPEKE) [74] is considered as an extension. Thus, a weak shared secret is used to create a stronger session key between two or more entities. This extension is optional for low-end cores but highly recommended for mid-end and high-end cores allowing them to enhance the security of data exchanged on the CAN communication channel.

Considering the current threats in the automotive networks emphasized by authors in

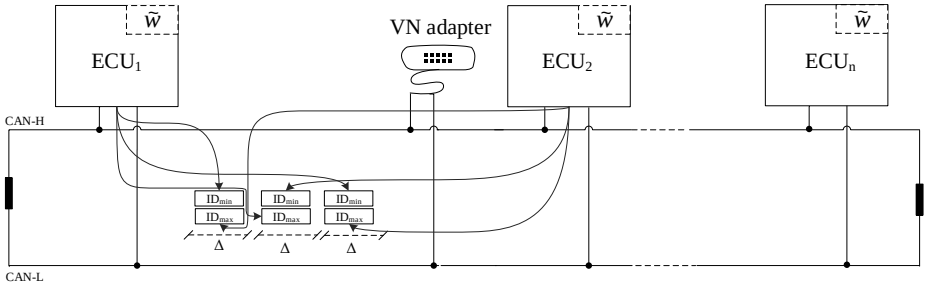


Figure 3.7: Structure of the CAN bus and addressed scenario for time-triggered key exchange

[2], [75], [76], there are various proposals from other works with methods to mitigate them. Since data is transmitted in Controller Area Networks using frames, proposals that include message authentication codes (MACs) are common [63], [77]. Other works provide mechanisms to optimize CAN frame transmission using different allocation methods [78], [79], [80] or present the safety and security integration trade-offs [81]. Even though methods based on message authentication codes are proven effective, there is little research done on key exchange mechanisms considering that MACs require a shared secret key to be shared between the parties. This is far from an easy topic because utilization of well-known methods such as RSA [58] or Diffie-Hellman [82] is difficult to be implemented due to the limited data field payload of CAN messages.

The method proposed by Mueller and Lothspeich [69] is somehow revolutionary for the Controller Area Networks because it uses the logic-AND property of the physical layer and is generally applicable to the protocol. Its extension done by Jain and Guajardo [70] provides an example of its utility to exchange secrets, e.g., cryptographic keys, on Controller Area Networks. Probing attacks [83] are possible on this type of key exchange setup. They were evaluated by the authors in [84], which show how masking unique node signaling behavior mitigates the effects of the probing attacks.

In our research work [29], discussed in this chapter, several key exchange mechanisms are proposed with the use of entire frames and the wired-AND property of the physical layer of the CAN bus. For ensuring resilience to probing attacks, the options are either to use the parallel transmission of frames as proposed by authors in [84] or to add random time delays to vary the clock drifts. In this way, the transmitter cannot be identified directly by an adversary. The physical layer has been proposed for key exchange in wireless networks [85], a well-known communication channel that is also used in modern vehicles.

3.2.2 Adversary model

In order to evaluate the security of proposed algorithms, a Dolev-Yao adversary [86] with full access to the Controller Area Network communication channel is considered. The first algorithms are not secured against this adversarial type since a legitimate node can be replaced from the vehicle bus without breaking the key exchange algorithm. The algorithm proposed by Mueller and Lothspeich [69] is vulnerable to this type of adversary as well. However, since it is hard to replace legitimate ECUs inside a vehicle without compromising vehicle level functionalities, such attacks are not foreseen to be easy to perform. Other types of attacks, such as bus-off attacks [66], will not remove legitimate nodes from the bus, but they will keep them idle for some time, until they recover from the bus-off state. Similarly, DoS attacks would compromise the key exchange protocol since the genuine nodes will end up using wrong (modified) keys for cryptographic methods such as message authentication codes (MACs). For weak adversaries that can only eavesdrop the communication from the channel, the proposals from [29] are considered to be secure. For stronger adversaries, an extension of the protocols is required in order to exchange the session keys using guessing resilient protocols such as Encrypted Key Exchange (EKE) [73], [87] or SPEKE [74] protocols with elliptical curve cryptography support and the changes proposed by [88].

3.2.3 Embedded platform setup and implementation considerations

For performing the experiments, Infineon development kits with automotive grade microcontrollers from the AURIX family were used. These boards are connected in pairs of two to a vehicle bus interfaced to a computer by a VN hardware equipment. The VN hardware equipment is developed by Vector and was used to log the communication channel for determining the arrival time for frames on the physical bus. The experimental setup is shown in Figure 3.8.

There are three microcontrollers which were used in the experiments:

- Infineon AURIX TC224, a single-core microcontroller with a maximum operating frequency of 133MHz, an internal RAM memory of 96kB and internal FLASH memory of 1MB,
- Infineon AURIX TC277, a triple-core microcontroller with a maximum operating frequency of 200MHz, an internal RAM memory of 472kB and internal FLASH memory of 4MB,
- Infineon AURIX TC297, a single-core microcontroller with a maximum operating frequency of 300MHz, an internal RAM memory of 728kB and internal FLASH memory of 8MB.

The Vector VN hardware equipment was connected to the CANoe tool from the laboratory PC and to the network bus for recording the bus traffic in real-time as a basis for the evaluation that was performed using the Mathematica tool.

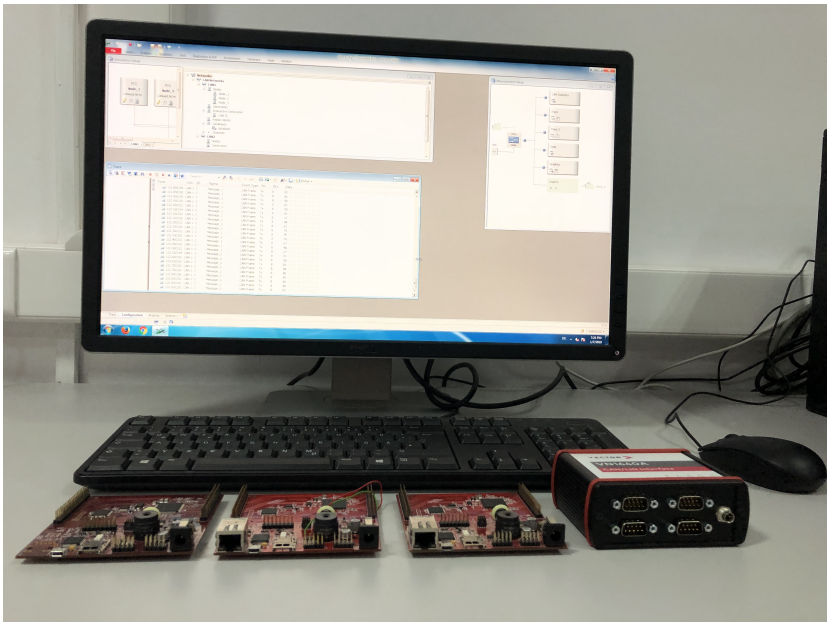


Figure 3.8: Hardware and software components employed for the experiments

Table 3.5: Summary of notations for time-trigger key-exchange protocols

ID	identifier field of a CAN message
ECU	Electronic Control Unit
Pr_{bad}	probability that a frame discloses the corresponding bit
$\mathsf{H}_{average}$	mean entropy (as function of k frames)
T	duration of the key exchange (as function of k frames)
Δ	estimated delay between two frames
$b_{\blacklozenge}$	array of random bits generated by $\blacklozenge \in \{A, B\}$
$\neg b_{\blacklozenge}$	complement of values in previous bit array $\blacklozenge \in \{A, B\}$
$x_{\blacklozenge}$	random delay generated by $\blacklozenge \in \{A, B\}$
$id_{\blacklozenge}$	array of random ID generated by $\blacklozenge \in \{A, B\}$
$\mathsf{ID}_{\mathsf{min}}$	the ID with the minimum value (whenever two IDs occur at the same time)
$\mathsf{ID}_{\mathsf{max}}$	the ID with the maximum value (whenever two IDs occur at the same time)

In Table 3.5, all notations from the following paragraphs are summarized. The notations are used in the equations that formalize the algorithms or their performance. For evaluating the performance of each algorithm, there are three characteristics measured during the experiments. The first is Pr_{bad} which is the probability that a frame is bad, which means it can be used by an adversary to extract a bit of the session key. The second characteristic is the mean entropy which is denoted as $\mathsf{H}_{average}$ and depends on the frames sent by each genuine node. The third characteristic is the time it takes to execute the entire key-exchange method, denoted as T.

The implementation details for the algorithms are discussed in what follows. In order to synchronize the transmission of frames at the same time between nodes, micro-second

level timer interrupts are implemented using the T13 timer from Capture/Compare Unit 6 Timer (CCU6) module from the AURIX microcontrollers. For ensuring communication capabilities on the Controller Area Network, the MultiCAN+ drivers are used. The drivers can store message objects with up to 64 different mailboxes. The bit rate for the CAN communication channel was set to 500Kbit/s as recommended by the SAE J2284-3 [89] standard for automotive networks. As the initial step for each algorithm, after microcontroller startup and the initialization part, each device expects a CAN frame that triggers the timer initialization with a timer interrupt period configured at $2\mu s$. When the interrupt is triggered, the node decides whether to send a remote or standard CAN frame. The time varies for each algorithm, i.e., frames are transmitted at the same time by both nodes, or they are transmitted at random timestamps so they do not overlap. After sending the message, each node saves the transmitted ID, timestamp for transmission time and reception time for each CAN frame sent by the other node. After all frames are transmitted and received, each node either computes the key using the random bits based on the timestamps for frames received from the other nodes or they compute the key using random bits based on the timestamps from all frames exchanged on the bus.

3.2.4 Data vs. Remote frame negotiation

The first algorithm follows the principle proposed by Mueller and Lothspeich [69] but requires frames to be exchanged on the bus between nodes instead of bits. The principle is explained in what follows, since it is also similar to other algorithms that are later described. Both nodes generate a random number of k bits, i.e., $b_A = b_A^1, b_A^2, ..., b_A^k, b_B = b_B^1, b_B^2, ..., b_B^k$. If the generated bit is 0, the node transmits a data frame, otherwise it sends a remote frame. Both nodes will start transmitting frames at the same time at intervals with the same period denoted as Δ. The time interval value, i.e., Δ, needs to be chosen to accommodate at least one frame and at most two frames for the chosen bitrate of the CAN bus. Time intervals of $200\mu s$ were used, but that can also be reduced to the maximum duration of two frames that have no data bytes, for the selected bitrate.

The possible collisions on the communication channel are shown in Figure 3.9. If the bits are complementary, i.e., 01 transmitted by ECU_A and ECU_B in (i) or 10 transmitted by the same ECUs in (ii), a data frame will be transmitted on the bus. The node that transmits a remote frame will see that a data frame is actually sent on the bus, but the node that transmits a data frame will have no information regarding what frame the other node had sent, data or remote. This is visible as (ii) or (iii) from ECU_B's perspective. This is later clarified by the transmission of the dominant bit complements shown in (v) and (vi), where either remote frames or data frames are transmitted on the bus as 10 and 01, opposite to (i), (ii) from the perspective of ECU_A and ECU_B. In the case of (iv), the bit value is compromised since an adversary will know that both nodes exchange a bit with the value of 1, since a remote frame is successfully transmitted on the communication channel. The same frame identifier is used by both nodes in all experiments performed for

evaluating the data vs. remote frame negotiation algorithm. For cases (iii) and (iv), Pr_{bad} is considered to be 0.5 since half of the bits exchanged by nodes can be eavesdropped by an adversarial node. This means that only half of the frames contribute to the key entropy, so the $\mathsf{H}_{average} = (1 - \mathsf{Pr}_{bad}) \times k$ is equal to $k/2$ where k is the number of frames. The time required to exchange the key between the nodes is twice the time k frames are transmitted, so $\mathsf{T}(k)$ is $2 \times k \times \Delta$. The experimental results for 32 frames exchanged by the nodes on the communication channel are shown in Figures 3.10, 3.11 and 3.12. The IDs for frames transmitted by both nodes where remote frames are shown with the negative value of the data frame IDs are shown in Figure 3.10. The overlaps that are visible on the CAN bus with associated timestamps are shown in Figure 3.11, while Figure 3.12 depicts that all frames arrived at the expected time for each Δ interval.

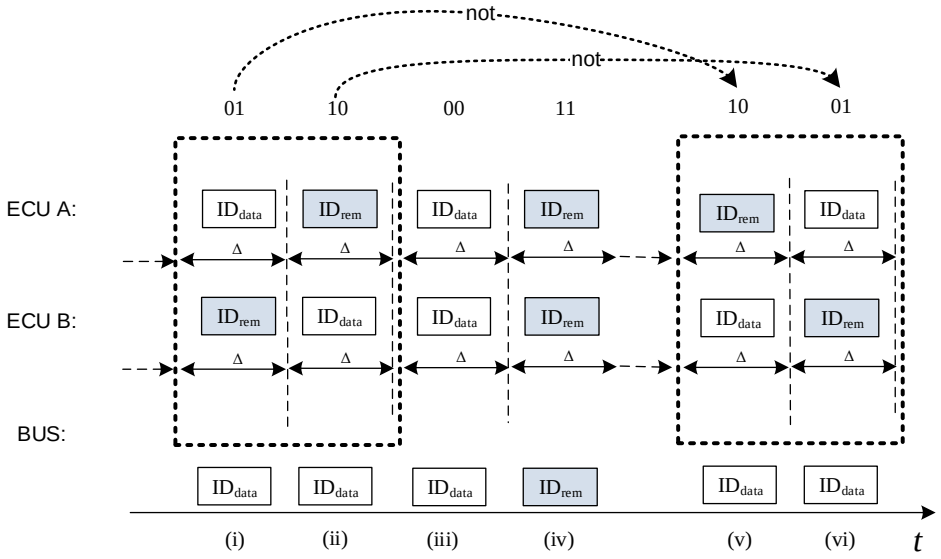


Figure 3.9: Data vs. Remote frame negotiation (based on Mueller and Lothspeich principle [69])

3.2.5 Minimax negotiation

This algorithm relies on random identifiers transmitted for each frame. Both nodes, ECU_A and ECU_B, generate a random number k of identifiers, $id_A = id_A^1, id_A^2, ..., id_A^k$, $id_B = id_B^1, id_B^2, ..., id_B^k$. In case the generated random identifiers are equal for one iteration, only one frame will be visible on the bus and the adversarial node can extract that bit

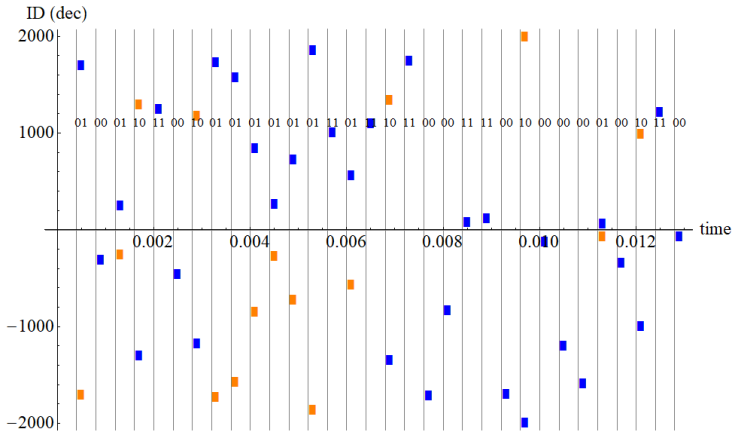


Figure 3.10: Data vs. Remote frame negotiation: frames on node A and node B (orange vs. blue)

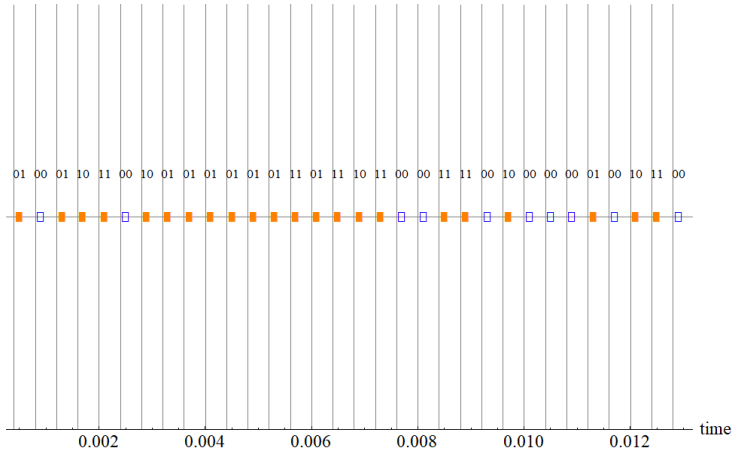


Figure 3.11: Data vs. Remote frame negotiation, frames arrived on the bus data (square) vs. remote (empty square)

from the session key. If the random identifiers are different, the identifier with the smaller value denoted $\mathsf{ID}_{\mathsf{min}}$ is the first to be transmitted on the bus, followed by $\mathsf{ID}_{\mathsf{max}}$ due to CAN arbitration. So, there is a probability, Pr_{bad}, that the randomly generated IDs are the same for one iteration of $1/2^{11}$ when using standard identifiers, i.e., 11 bit size, and $1/2^{29}$ when using extended identifiers, i.e., 29 bit size. This means 1 out of 2048 possibilities for standard identifiers and 1 out of more than 500 million chances for extended identifiers, which is a very low probability. One measure to avoid this types of collision is to include random bytes in the data field. This method, if used, adds more time to the frame duration and requires an increased value for Δ. During the experiments, frames

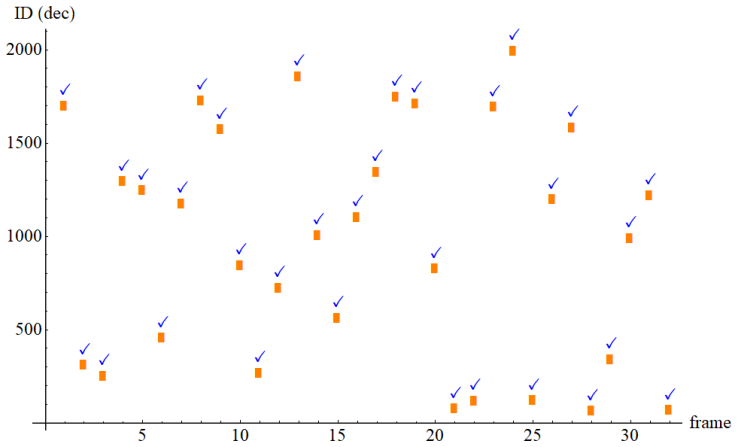


Figure 3.12: Data vs. Remote frame negotiation: expected arrival frame (as checkmark) vs. arrived frame (square)

without bytes in the data field are used, while the value of Δ is set to $200\mu s$ and to $125\mu s$. This allows two frames to be transmitted in a $2 \times \Delta$ interval without any overlaps. In the first part of Figure 3.13 represented as (i), there are two identical frames transmitted on the bus denoted as 00, since their identifiers are equal. For the following parts of Figure 3.13, (ii) and (iii), there are different frame identifiers transmitted by the nodes with the lower value, $\mathsf{ID}_{\mathsf{min}}$, being transmitted on the bus first, followed by $\mathsf{ID}_{\mathsf{max}}$. This is denoted as 01 and 10. Except for (i) where there is only one frame transmitted on the bus, for (ii) and (iii) both frames are transmitted in a $2 \times \Delta$ interval which encompasses two frames with a maximum possible duration of $212\mu s$ for a baud rate of 500Kbps. In case there are no equal identifiers, there is a 1 bit entropy that results for each transmission pair extracted by the receivers that is based on the frame transmission order on the bus. For $2k$, frames there is a mean entropy equal to k, i.e., $\mathsf{H}_{average}(k) \approx k$. Considering that each node transmits k frames but within a $2 \times \Delta$ interval at a time, the key exchange duration is of $\mathsf{T}(k) = 2k\Delta$. The experimental results for 64 frames with a standard identifier exchanged by the nodes on the communication channel are shown in Figures 3.14, 3.15, 3.16. In Figure 3.14, the IDs for frames transmitted by both nodes are shown where frame identifiers are shown with blue and orange squares. The frame transmission sequences visible on the CAN bus with associated timestamps are shown in Figure 3.15 while Figure 3.16 shows that all frames arrived at the expected time for each Δ interval.

3.2.6 Time-triggered Minimax negotiation

Similarly to the Minimax Negotiation, this algorithm relies on k random identifiers for each frame $id_A = id_A^1, id_A^2, ..., id_A^k$, $id_B = id_B^1, id_B^2, ..., id_B^k$ but also on a second random sequence of k bits generated by each node, $b_A = b_A^1, b_A^2, ..., b_A^k, b_B = b_B^1, b_B^2, ..., b_B^k$.

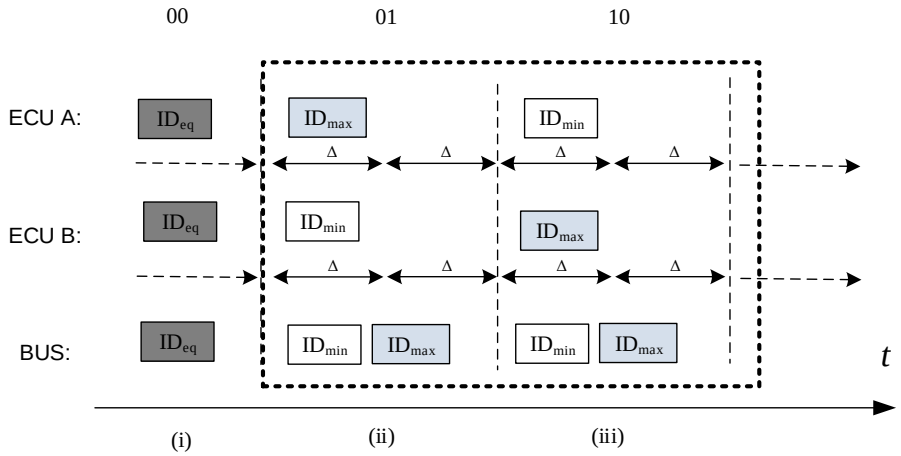


Figure 3.13: Minimax frame negotiation: principle

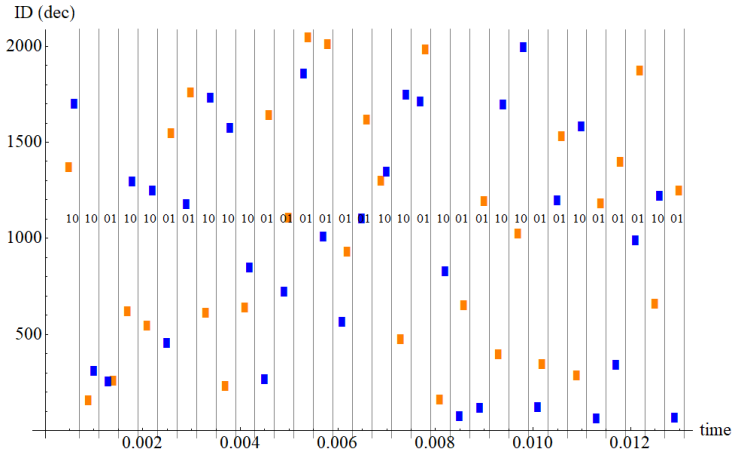


Figure 3.14: Minimax frame negotiation: frames on nodes A and B (orange vs. blue)

The time slots in which frames are expected to be sent are $(2i + bi) \times \Delta$, which, depending on the value of each bit would result in $2i \times \Delta$ or $(2i + 1) \times \Delta$. For a specific time slot, in case the bit value is different for the nodes, i.e., 10 or 01, both are extracted by the nodes without revealing the information to an adversary. If the bit value is the same, i.e., 00 or 11, bits are revealed to the adversary since the frame is transmitted one after the other with a very low inter-frame space compared to the 10 and 01 cases. This means

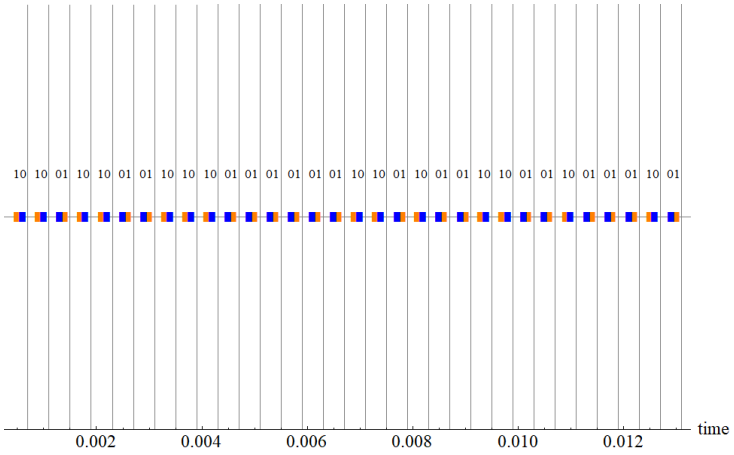


Figure 3.15: Minimax frame negotiation: frames arrived on the bus from Node A (orange) vs. Node B (blue)

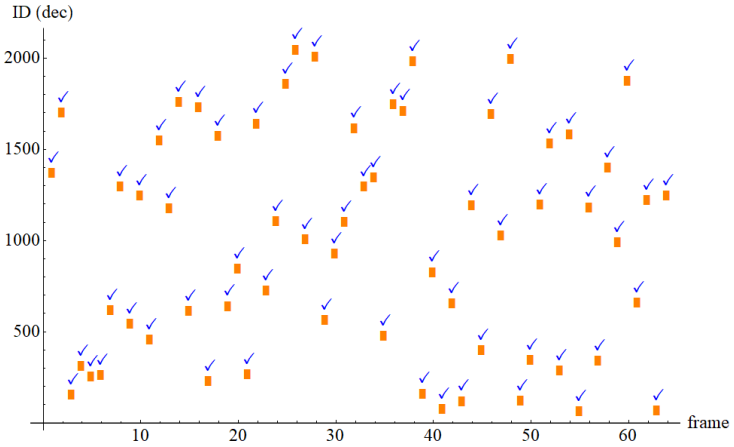


Figure 3.16: Minimax frame negotiation: expected arrival frame (as checkmark) vs. arrived frame (square)

that, in half of the possible scenarios, an adversary will extract the bits shared by the nodes, so the value of Pr_{bad} is $1/2$ while the average entropy $\mathsf{H}_{average}(k)$ is equal to half of the iterations, $k/2$. In order to prevent overlaps between consecutive iterations, the total time of one iteration is updated to $3 \times \Delta$, which leads to a total key exchange time of $\mathsf{T}(k) = 3k\Delta$. In Figure 3.17 the 10, 01 and 11 scenarios are shown, omitting the 00 case, because in that case, the bits from the key-exchange protocol can be extracted by an adversary. The experimental results for 64 frames exchanged by the nodes on the communication channel with the value of Δ set to $200\mu s$ are presented in Figures 3.18, 3.19

and 3.20. In Figure 3.18, the IDs for frames transmitted by ECU_A and ECU_B are shown with frame identifiers represented with blue and orange squares. The frame transmission sequences visible on the CAN bus with associated timestamps are shown in Figure 3.19, while Figure 3.20 shows that all frames arrived at the expected time inside each $3 \times \Delta$ interval. In case the random IDs are different between nodes ECU_A and ECU_B for a specific time slot, by combining this algorithm with the Minimax negotiation, the average entropy $\mathsf{H}_{average}(k)$ can be increased from $k/2$ to k. The rationale is that the frames are transmitted by both nodes and the frame with the lower ID value $\mathsf{ID}_{\mathsf{min}}$ will win the arbitration with $\mathsf{ID}_{\mathsf{max}}$, in case the ID values are different. So, the resulted entropy, is the sum of entropies from both Minimax and Time-triggered Minimax protocols.

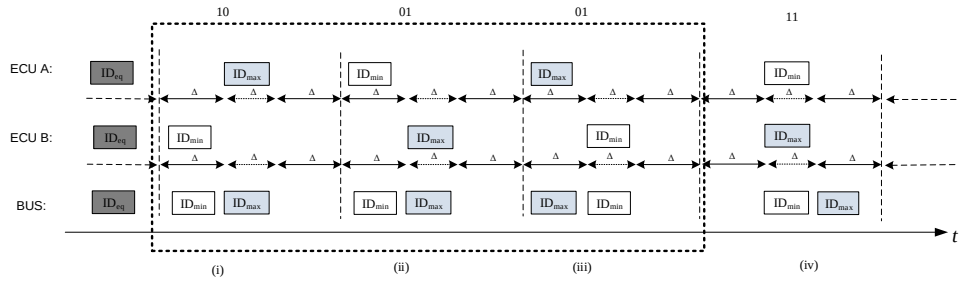


Figure 3.17: Time-triggered Minimax frame negotiation: principle

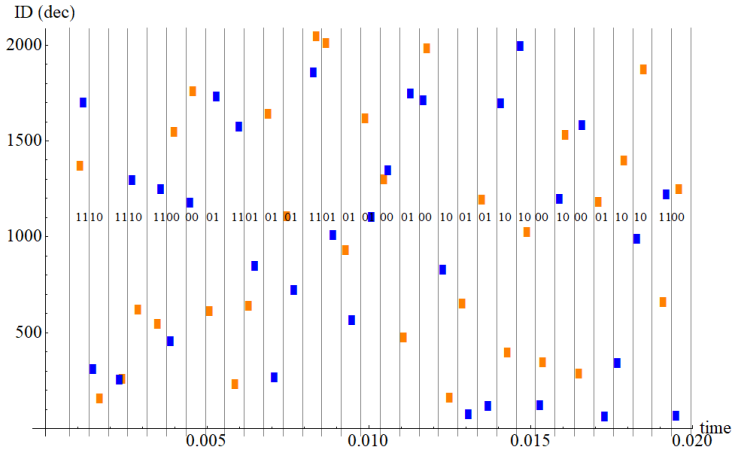


Figure 3.18: Time-triggered Minimax frame negotiation: frames on node A and node B (orange vs. blue)

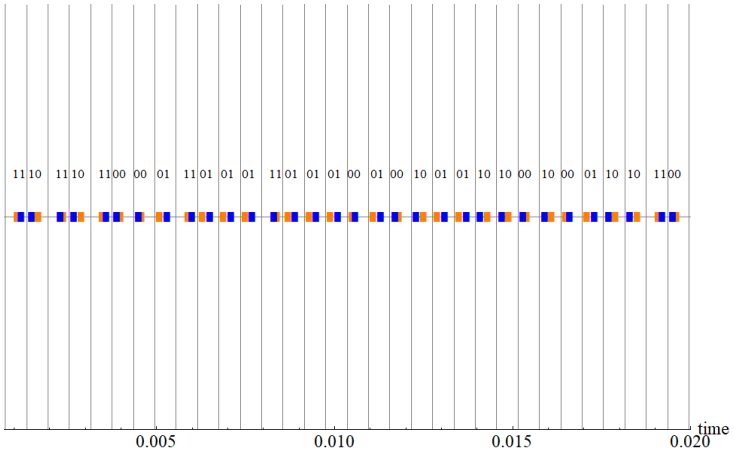


Figure 3.19: Time-triggered Minimax frame negotiation: frames arrived on the bus from Node A (orange) vs. Node B (blue)

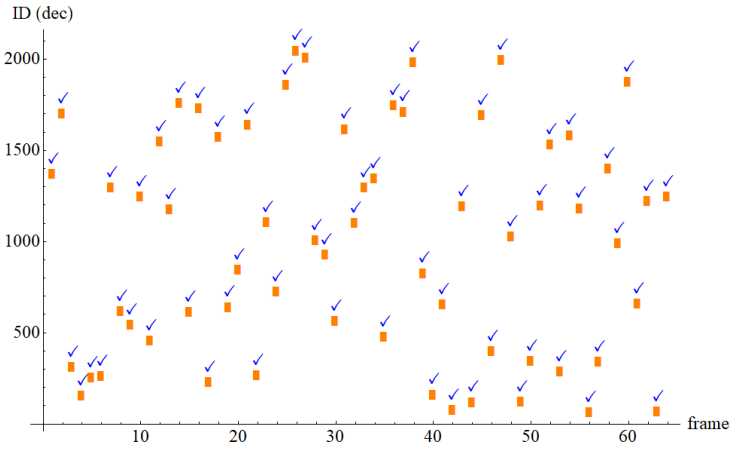


Figure 3.20: Time-triggered Minimax frame negotiation: expected arrival frame (as checkmark) vs. arrived frame (square)

3.2.7 Randomized time-triggered key-exchange

This algorithm relies on random values generated by both nodes for selecting random time slots when to transmit frames on the bus. Considering that each node generates k random values, $x_A^1, x_A^2, ..., x_A^k$ for ECU_A and $x_B^1, x_B^2, ..., x_B^k$, for ECU_B with k chosen as part of interval $[1..\ell]$, frames are transmitted by each node at time $x^i \times \Delta$. When ECU_A transmits a frame, ECU_B considers it as 0 and when ECU_B transmits a frame ECU_A will consider it as 1. The values chosen for the experiments are $\Delta = 200\mu s$

and $\ell = 512$. In Figure 3.21, the randomized transmission scenarios are shown where both ECU_A and ECU_B broadcast frames at random Δ intervals. The time difference between consecutive frames transmitted by ECU_A and ECU_B are emphasized in the same figure. The experimental results for 64 frames exchanged by the nodes on the communication channel are shown in Figures 3.22, 3.23 and 3.24. Figures 3.22 and 3.23 show that first 64 frames arrived at the expected time. Since the time slots for transmission are randomly generated by each node, it can be anticipated that nodes can also transmit frames in the same time slot. For the first 64 frames, there was only one such occurrence marked with X in Figure 3.23. There were multiple collisions during the entire experiment marked with X in Figure 3.24, where all transmission times for corresponding frame identifiers are shown. The probability for one collision to occur on the bus, Pr_{bad}, is k/ℓ because the probability of having time slots with no collision is $(\ell - k)/\ell$ for k selected slots. This means that an adversary can extract the bits from the bus in k/ℓ situations, so the collision probability is k/ℓ. Since ECU_A transmits k frames and there are $2k$ frames on the bus, there are $(2k)!/k!$ possible combinations of frames transmitted on the bus by ECU_A. The entropy estimated for this protocol can be computed based on the total possible combinations divided by $k!$, since the frame order is also random. This means that $\mathsf{H}_{average}(k) = -\log_2\left((k!)^2/(2k)!\right)$. Considering that an average number of frame collisions is of k^2/ℓ, ℓ can be increased in order to maximize the number of correct frames transmitted on the bus without collisions, that is $k - k^2/\ell$. A graphical representation of average entropy that can be obtained by using Randomized time-triggered key exchange is shown in Figure 3.25. If the values of k and ℓ are higher, the entropy value is also higher, but the time duration for the entire algorithm is increased. The total time for transmitting all frames, so for the randomized time-triggered key exchange, is $\mathsf{T}(k) = \ell\Delta$.

A summary of Pr_{bad}, $\mathsf{H}_{average}(k)$ and $\mathsf{T}(k)$ for the earlier described algorithms is presented in Table 3.6. The information for each algorithm is based on the experimental data. Frames were captured from the bus during the experiments using a Saleae logic analyzer.

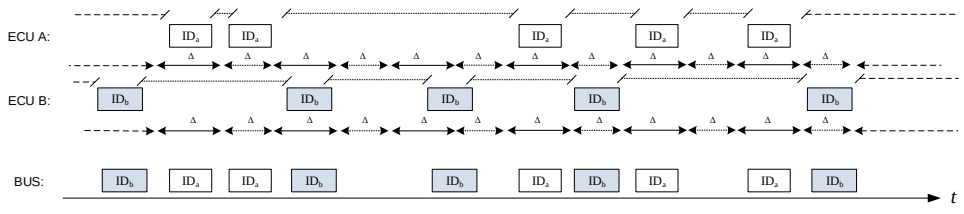


Figure 3.21: Randomized time-triggered key exchange: principle

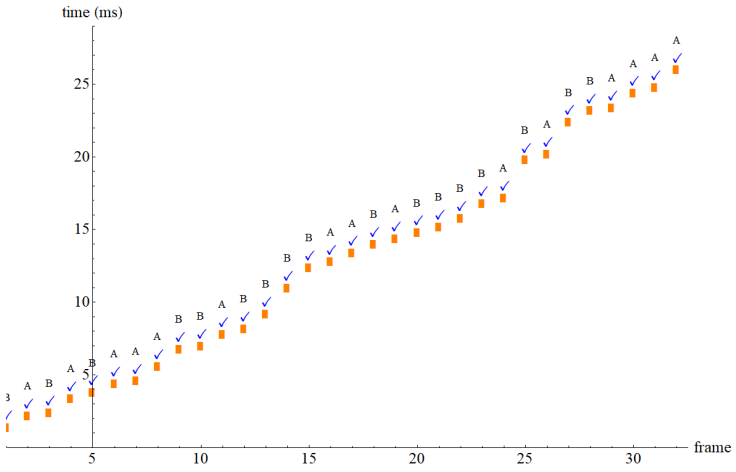


Figure 3.22: Arrival time for the first 32 frames with Randomized time-triggered key-exchange

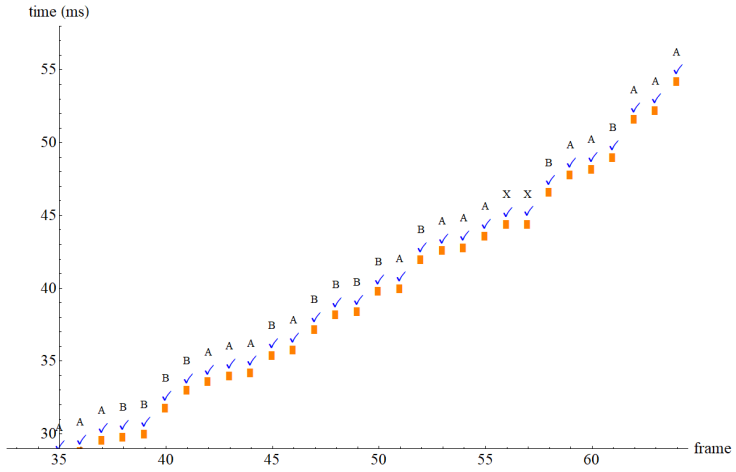


Figure 3.23: Arrival time for the last 32 frames with Randomized time-triggered key-exchange

3.2.8 Extension of the key negotiation protocols

An improved version of the Time-Triggered Minimax algorithm is shown in what follows using the Diffie-Hellman (DH) version [82] of the Encrypted Key Exchange (EKE) protocol [73], which allows us to bootstrap a high entropy session key on a lower entropy shared secret. Then, an extension of the improved version is shown, so a group session key can be shared by all nodes from the communication channel.

Piggy-backed Diffie-Hellman EKE/SPEKE. Even though the proposed algorithms

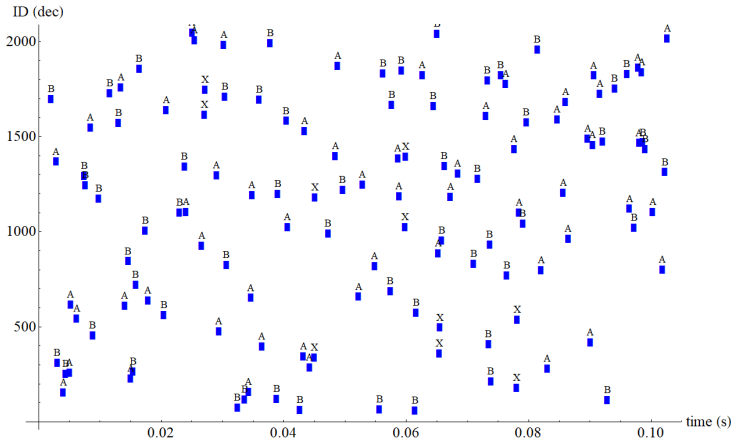


Figure 3.24: Frame arrival time and ID value with Randomized time-triggered key-exchange

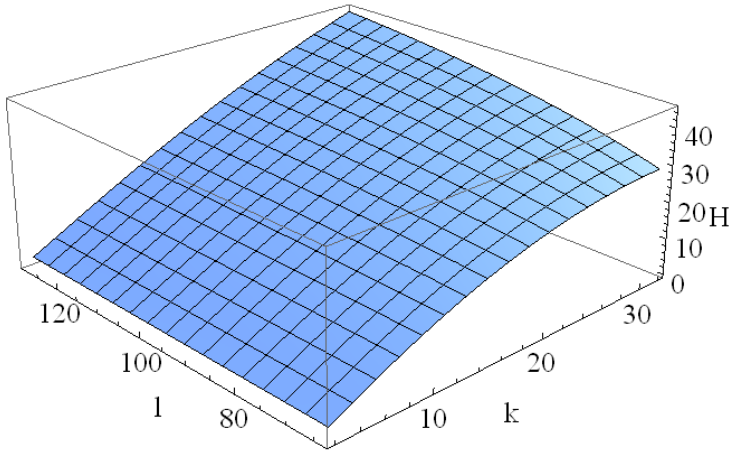


Figure 3.25: Entropy extracted from the Randomized time-triggered key-exchange (approximation)

rely on software implementation and do not require additional changes to the hardware compared to the work from [69], there is a low efficiency for data and transmission times since a covert bit exchange requires an entire CAN frame. But this drawback is changed as an opportunity to piggyback the CAN frames with parts of Diffie-Hellman-based [82] key shares of EKE [73]. This improvement is resilient to guessing attacks, as shown by authors in [87]. In order to reduce the memory and time required for the computation, the EKE [73] is updated to use elliptic curves by following the SPEKE protocol [74]. The backbone for the EKE-DH frames can either be the Time-Triggered Minimax negotiation

Table 3.6: Comparison of the key-exchange schemes

	Data vs. Remote	Minimax	T-T Minimax	Randomized T-T
Pr_{bad}	$1/2$	$1/2^{11}$	$1/2$	k/ℓ
$\mathsf{H}_{average}$	$k/2$	k	$k/2$	$-\log_2\left(\frac{(k!)^2}{(2k)!}\right)$
T	$2k\Delta$	$2k\Delta$	$3k\Delta$	$\ell\Delta, \ell > k$

or the Randomized Time-Triggered key exchange, because both nodes already know their share of the key before starting the key exchange. The bits used for EKE-DH are piggy-backed to the frames and illustrated in what follows using the Time-triggered Minimax algorithm. Extraction of the session key is performed only after the key exchange is finished.

Time-triggered Minimax with piggy-backed EKE/SPEKE. The following parameters are defined for the protocol: an elliptic curve $E/\mathbb{F}_q$, a fixed point $P = f(\widetilde{w}) \in E(\mathbb{F}_q)$ of prime order p and the time slot delay Δ. In this case, f maps the key exchanged using the algorithm to a point P on the elliptic curve. The key that is exchanged is the same as a generator g in the SPEKE protocol [74]. Both nodes follow the steps that are defined below:

1. $\mathsf{Setup}(1^k)$ where, ECU_A and ECU_B based on, two random arrays of bits, $b_A = b_A^1, b_A^2, ..., b_A^k$, $b_B = b_B^1, b_B^2, ..., b_B^k$, both with k elements, two arrays of random IDs, $id_A = id_A^1, id_A^2, ..., id_A^k$, $id_B = id_B^1, id_B^2, ..., id_B^k$, with k elements and two positive integers x_A and x_B compute $s_A = b_A x_A P = \{s_A^1, s_A^2, ..., s_A^k\} \in E(F_q)$, $s_B = b_B x_B P = \{s_B^1, s_B^2, ..., s_B^k\} \in E(F_q)$ as shareable points from the elliptic curve,
2. $\mathsf{SendCyclic}(i), i = 1..k$ where each node $\mathrm{ECU}_\alpha, \alpha \in \{A, B\}$ transmits k frames with the identifier id_α^i and s_α^i as payload in the $3i\Delta, i = 1..k$ time slots following the time-triggered Minimax negotiation algorithm,
3. $\mathsf{ExtractSessionKey}(T)$ where both nodes ECU_A and ECU_B recover the key based on the time-triggered Minimax protocol using the timestamps for the received frames $T = \{(id_1, t_1), (id_2, t_2), (id_3, t_3),..., (id_{2k}, t_{2k})\}$ and then extract session key checking if $|t_{2i-1} - t_{2i}| < \epsilon, i = 1..k$ and setting $b_\beta^i = \neg b_\alpha^i$ or to $b_\beta^i = b_\alpha^i$, having the shared session key computed as $K_{ses} = x_\alpha b_\alpha^{-1} Q$ where Q is the point from the elliptic curve recovered from k frames,
4. $\mathsf{ConfirmSessionKey}(K_{ses})$ where nodes transmit frames containing the MAC (message authentication code) computed using the shared session key over the shareable points from the elliptic curve transmitted by each node in step 2, $s_\alpha, \alpha \in \{A, B\}$. After receiving the frames, both nodes compute the MAC locally over $s_\alpha, \alpha \in \{A, B\}$ using the extracted session key K_{ses} and verify the correctness of the received MAC. After confirmation, the key recovered based on the Time-Triggered Minimax protocol can be updated to $\widetilde{w} = MAC_{\widetilde{w}}(K_{ses})$,
5. $\mathsf{Abort}(T)$ where any of the nodes abort the key exchange protocol by transmitting

error frames due to any of the following conditions: (1) more frames than expected ($2k$) are received during step 2, (2) the frames are not sent in pairs so they do not follow the time-triggered Minimax protocol, (3) one of the nodes does not send any frame inside a $3 \times \Delta$ time slot.

The final part of the protocol consists of the confirmation of session keys based on the verification of the MAC. An elliptic curve defined over a 160-bit field is chosen, so the X coordinate has a size of 20 bytes. One additional byte that contains the sign for the point is also required. Thereby, in the experiments, k = 21 was used so there are 42 frames transmitted by the nodes.

Security arguments. The analysis for the security of the SPEKE protocol was out of scope in [29] since this has already been done by others in research works [90], [91] and [88] and it is already used in standards such as IEEE 1363.2 [92] and ISO 1170-4 [93]. Although impersonation attacks in parallel sessions or key malleability attacks are deemed possible against SPEKE [74] by authors in [88], these types of attacks are not applicable to the proposed methods and would not affect the disclosure of the weak secret. The vulnerabilities for these attacks can be solved as shown in the research works that analyze the SPEKE security [90], [91] and [88]. The counter-measures proposed for existing vulnerabilities can be applied to this protocol considering that SPEKE [74] can be used to increase the entropy of the weak secret $\widetilde{w}$ from the Time-Triggered Minimax negotiation protocol. Considering a man-in-the-middle attack scenario, an adversary cannot derive the point $P = f(\widetilde{w})$ on the elliptical curve because of the unknown weak shared secret. Thus, the adversary would generate a different point P' that, in the context of the piggy-backed frames, results in $b_{adv}x_{adv}P'$ as transmitted payload with $P' \neq P$. This leads to a different session key between the genuine node, $x_\alpha x_{adv}P'$, and the adversarial node α and $x_{adv}x_\alpha P$, because of the different points, P and P'. In case of probing attack scenario, an adversary is unable to impersonate a genuine node unless he learns the weak secret from the Time-Triggered Minimax negotiation and then replaces the node on the communication channel before the piggy-backed encrypted key exchange protocol is performed. This type of attacker is referenced by the authors from [69], but this attack is hard to perform on in-vehicle networks since it requires physical access to them. By following the resurrecting-duckling paradigm [71], the assumption is that all nodes connected to the in-vehicle network have a weak shared secret from the factory or from the first operating power cycle inside the vehicle. Using this secret, the nodes are able to generate a stronger session key using the SPEKE protocol [74]. Timings for time slots, total duration of key exchange and entropy for each protocol are shown in Table 3.7.

Multi-party version of the scheme. All the proposed key exchange protocols using the Diffie-Hellman [82] method or SPEKE [74] can be extended from two nodes to multiple nodes as already discussed by authors in research works such as [94] or [95]. An alternative to the one presented in [94] is described in what follows. All ECUs, i.e., $\mathrm{ECU}_i, i = 1..n$, communicating on the CAN bus can be grouped in pairs. This means

Table 3.7: Summary of experimental parameters and results

	Data vs. Remote	Minimax	TT Minimax	Randomized TT	SPEKE TT-Minimax
$\Delta\ (\mu s)$	200	125	200	200	200
$H_{average}$ (bits)	32	64	32	124	160
T (ms)	25	16	38	102	12 (bus) + 98 (comp.)

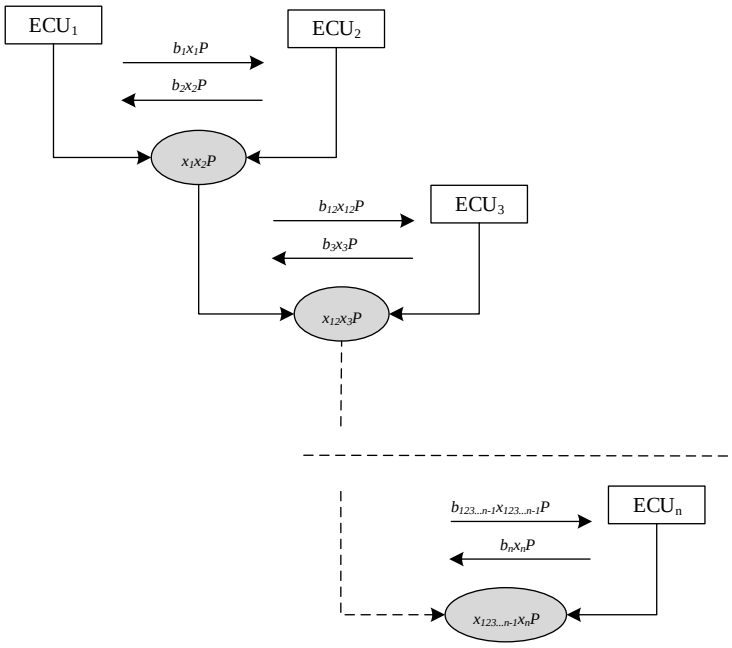


Figure 3.26: Suggested group key exchange with EKE-DH

that if a session key, i.e., x_1x_2P, has been shared between ECU_1 and ECU_2, both can generate a new point on the curve by applying the key derivation function on the session key and providing it as input to the pseudo-random number generator (PRNG). This will result in the $b_{12}x_{12}P$ public key for the ECU_1-ECU_2 pair that can be exchanged with ECU_3 and a new session key can be shared between ECU_1, ECU_2 and ECU_3. This extension applies to any number of ECUs but requires the additional step of generating a common public key between k nodes before sharing a session key with the $k+1$ node. For a number of n ECUs, this will result in a shared key computed as $x_{123\ldots n-1}x_nP$. The session key exchange between n ECUs is illustrated in Figure 3.26 and shows each ECU in a rectangular box and the sessions key shared within each pair in gray ovals. The arrows show the contribution of each node inside a pair and the shared session key for each step. Any of the nodes inside a pair group of more than two nodes can take ownership of

the pair and continue the group key exchange on the communication channel. If multiple nodes from the same pair group transmit frames at the same time, this may lead to transmission errors flagged by error frames. One way to prevent this is to allow one node to perform the key exchange inside each pair group. In case ECUs have low computational resources that cannot use Diffie-Hellman [82] method or SPEKE [74], the group key exchange using the Time-Triggered Minimax negotiation and Randomized Time-Triggered key negotiation protocols. These key exchange methods rely only on the timings for the transmitted and received frames, without the EKE-DH part (which can be additionally used to bootstrap security based on low-entropy secrets when needed).

3.3 Concluding remarks

In this chapter, the evaluation of the libraries using the Infineon TC297 platform was presented. Automotive systems that use microcontrollers from the AURIX family can integrate elliptic curve cryptography in the applications they are intended for and can secure the traffic data transmitted on the vehicle bus. Based on the evaluation, the library with the fastest runtime for all operations is RELIC [49]. Support for securing high-level applications with C++ libraries that implements elliptic curve cryptography is offered by MIRACL [48]. Nonetheless, the library that is available to be used with a FIPS 140-2 [61] certification is wolfCrypt [59], separate or as part of wolfSSL [57]. Four time-triggered key exchange protocols, i.e., Data vs. Remote, Minimax, Time-triggered Minimax, Randomized Time-Triggered, were presented and evaluated on an experimental setup with respect to three main characteristics: the probability of exchanged bits to be revealed during the key exchange, the average entropy and the total duration of the key exchange. The first two characteristics were the same for Data vs. Remote and Time-triggered Minimax protocols but with an increased duration for the latter. The Minimax negotiation protocol has a lower probability compared to these algorithms of revealing the covert bits due to the utilization of 11-bit or 29-bit frame identifiers and a higher entropy. The Randomized Time-Triggered algorithm depends on the number of intervals that is selected for the key exchange. This means the key exchange will be executed with a higher duration than the other algorithms but with an increased entropy. An extension that combines two of the proposed algorithms with public key exchange methods was evaluated with the goal of increasing the entropy of the key exchange method. The multi-party variant of the key exchange algorithms was discussed.

Chapter 4

Time-Covert Authentication on the CAN Bus

This chapter is based on a previous research paper of the author [30], which presents several frame scheduling optimization methods that are necessary for improving the performance of time-covert channels for the Controller Area Networks. A shorter previous research work [24] was the foundation of the author's Master Thesis which also addresses time-covert channel design but without the optimizations from the later work. The evaluation on an embedded platform of the time-covert channel with frame scheduling optimization is due to the later work [30]. The data throughput and security level of the time-covert channel with frame scheduling optimization is compared with other proposals as well.

4.1 Background on CAN timings and related works

In one of the previous research works of the author [24], an authentication protocol is implemented based on a covert timing channel. The authentication protocol did not use the data field of the CAN message and was entirely controlled through timing delays using the hardware timers of a microcontroller. One limitation of this work is related to the performance of the authentication protocol when real-world CAN data is transmitted on the same network where the covert timing channel is implemented and which results in a very reduced covert bit rate. The authors from a different research work [96] also propose a covert timing channel for authenticating the transmitters, but the authentication rate is also reduced to only one covert bit for each frame.

One option for improving the performance of covert timing channels is to optimize the frame transmission times without reducing the communication channel busload. Optimization is based on time synchronization between nodes connected on the CAN bus. This is rather a challenging aspect since the Controller Area Networks do not have a spe-

cific clock line and nodes usually rely only on their local time base. Recent industry publications require time synchronization support for automotive communication networks [97], with a worst-case accuracy for CAN node reference clocks of $10\mu s$, as shown in section 4.1 from [98]. This means that the assumptions for implementing a time covert channel need to be aligned with the current requirements of the automotive industry considering the aforementioned specifications. The importance of time synchronization is also confirmed by the availability of CAN-based time-triggered communication protocols that were standardized [99] based on proposals from the past like Time Triggered CAN (TT-CAN) [100].

In case there is no traffic optimization, the inter-frame delays from a real-world vehicle trace are usually in the area of $20\mu s$ but can vary up to $4ms$ or more, as shown in Figure 4.1. The information from this figure is presented for a CAN bus where cyclic messages are transmitted by genuine nodes but they arrive at random times on the bus with certain delays, whenever arbitration occurs, reducing the inter-frame space whenever this happens. By optimizing the frame scheduling, specific inter-frame delays are predefined in order to follow a periodic pattern, as shown in Figure 4.2. The busload for both scenarios is around 30%-40%, which is common for CAN buses used in the automotive industry.

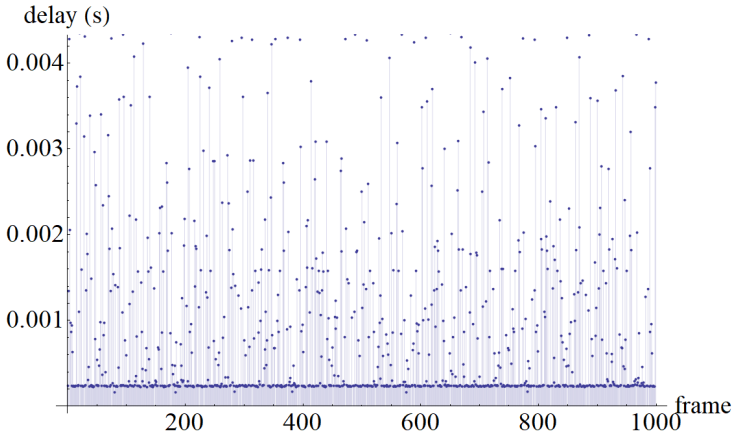


Figure 4.1: Delays between frames: real-world car at a busload of $30 - 40\%$

4.1.1 Clock skews and limitations in a previous work

In order to explain the limitations of a previous work [24], the first thing that needs to be discussed is how the clock skews accumulate and what is the main reason behind that. Thereby, the frames transmitted by three ECUs at a specific time interval, δ are shown in Figure 4.3. Due to internal clock imperfections, each node has a different view of δ which is noted as δ_1, δ_2 and δ_3 for ECU_1, ECU_2 and ECU_3. If ECU_1 measures the

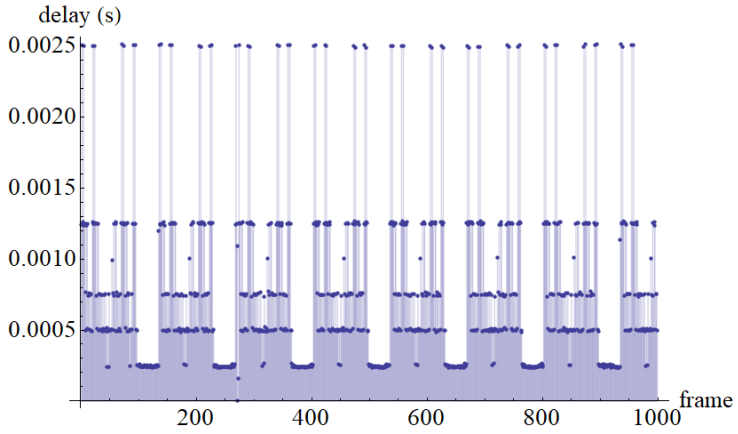


Figure 4.2: Delays between frames: optimized traffic on the setup at a busload of $30-40\%$

reception time for frames transmitted by ECU_2 and ECU_3, the delays caused by clock imperfections will accumulate. In Figure 4.3, the clock skew representation is shown for ECU_2 and ECU_3, as seen by ECU_1. The differences between δ_3 and δ_1 increase over time, so the slope for the skew is positive while the differences between δ_1 and δ_2 decrease in time, so the slope for the skew is negative. This means that the internal clock from ECU_1 is faster than the clock from ECU_2, but slower than the one from ECU_3.

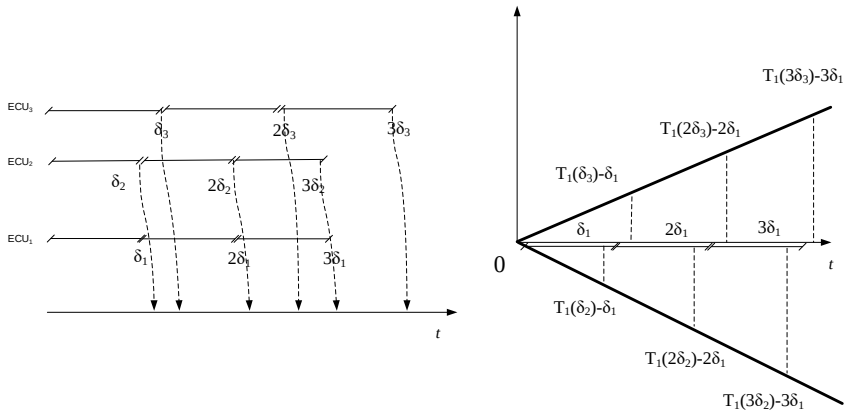


Figure 4.3: Accumulation of clock skews for ECUs broadcasting at interval δ

In Figure 4.4 from the previous work [24], the clock skews that are depicted are measured either by an Infineon board or a Vector VN adapter for frames that are transmitted by an Infineon TC277 microcontroller. The frame transmission is done follow-

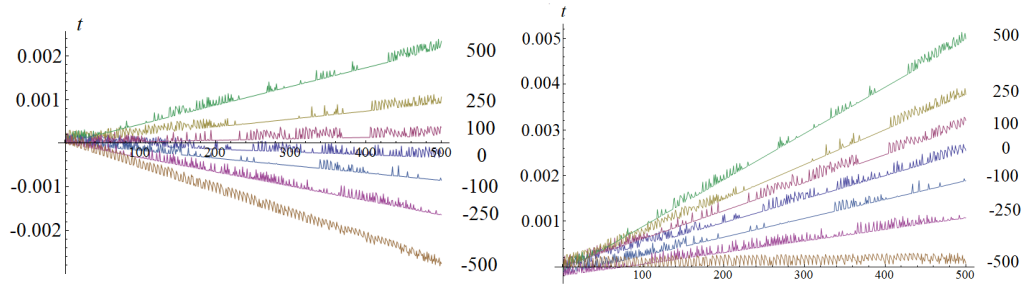


Figure 4.4: Skews for a frame sent from an Infineon TC277 as recorded by an Infineon board (left) and CANoe/VN CAN adapter (right) in [24]

ing a cycle time, but with delays, which are added based on timer ticks of $10ns$, set at $\pm 100, \pm 250, \pm 500$ which are visible in both sides of the figure. This leads to different slopes for each clock skew that depend on the delays and a different arrival time for the frames as seen by the Infineon node and the Vector VN adapter.

Unfortunately, the design from the previous work [24] is limited by the bus traffic that is not optimized which reduces the performance of the time-covert channel. This is shown in Figure 4.5 where, as presented on the left side, if the bus is free, no additional delays are visible. In the same figure, but on the right side, it is shown what happens when frames transmitted using the covert channel are delayed by un-optimized regular bus traffic. Additional delays caused by arbitration or a heavily loaded bus cause the receiver nodes to report false positives, failing to authenticate genuine frames on the time-covert channel that arrive late due to arbitration. One way to mitigate this limitation is to optimize frame scheduling on top of the time-covert channel, which will prevent arbitration between genuine frames and cause unexpected delays for the transmission and reception time.

4.1.2 Worst-case arrival times

An important aspect that needs to be discussed before optimizing the frame scheduling is the worst-case arrival time for frames transmitted on the CAN bus. Considering that ID-based arbitration from the Controller Area Networks allows frames with lower ID values to be transmitted first on the bus, frames with higher ID values will be transmitted with some delays, depending on the bus load. The computation for the busy period t and the worst-case queueing delay w are formalized for a CAN frame ID m based on the notations and definitions from [101]:

$$t_m^{n+1} = B_m + \sum_{k \in hp(m) \cup m} \left\lceil \frac{t_m^n + J_k}{T_k} \right\rceil C_k \tag{4.1}$$

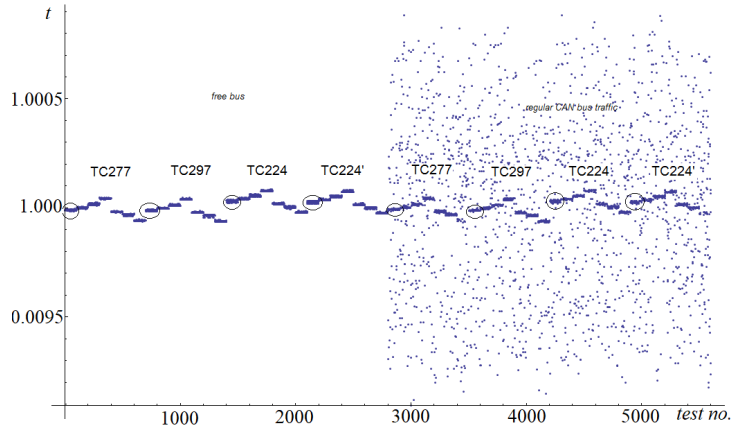


Figure 4.5: Forced delays as recorded in [24] for a free bus (left) vs. a bus with regular network traffic (right)

$$w_m^{n+1}(q) = B_m + qC_m + \sum_{k \in hp(m)} \left\lceil \frac{w_m^n + J_k + \tau_{bit}}{T_k} \right\rceil C_k \tag{4.2}$$

The Equations 4.1, 4.2 and notations from [101] are explained in what follows considering that m is the CAN frame ID that also defines its priority on the bus. In both equations, B_m is the blocking delay caused by frames with lower ID values to be transmitted before message m, J_k is the queueing jitter for the frame with ID k, T_k is the period for that frame, C_k is its worst-case transmission time, while $hp(m)$ is the list of messages with a lower ID value than m. Additional terms from the second equation are q, which is the instance of CAN frame m, C_m, which is the maximum transmission time for CAN frame m and τ_{bit}, which is the configured bit time for CAN communication. According to authors from [101], in order to solve the busy period t and the worst-case queueing delay w, the equations need to be solved for n frames until the same values are obtained for consecutive iterations, i.e., $t_m^{n+1} = t_m^n$ and $w_m^{n+1}(q) = w_m^n(q)$.

In order to determine the busy period t and the worst-case queueing delay w in the defined scenario, the equations are applied to 40 different IDs with cycle times of $10ms$, $20ms$, $50ms$ and $100ms$. The busload measured for the scenario is of 40%, quite common for real-world vehicles. The following values are set for the terms from the previous equations: $\tau_{bit} = 2\mu s$ since the CAN bus bit rate is of 500Kbps, $C_k = 270\mu s$ because the worst-case duration for a CAN frame is of $(55 + 10B)\tau_{bit}$ based on information from [101], where B is the payload size, and $B_m = 270\mu s$ except for the CAN frame with the lowest ID value. The busy period and worst-case queueing delay obtained for each ID in the scenario are shown in Figures 4.6 and 4.7. The blue circles from Figures 4.6 and 4.7 are for the normal case where there are only the 40 IDs transmitted. In case additional frames with authentication data are required for the original messages, i.e., the

number of transmitted frames is doubled, both the busy period and queueing delay will increase. For a $100ms$ cycle-time ID the busy period is of $12ms$ for normal traffic and up to $34ms$ in case additional frames with authentication data are sent. This means that the busload is problematic for all frames with a considerable negative impact on low-priority frames, i.e., frames with a high ID value that are more likely to lose arbitration on the bus. Thereby, optimizing frame allocation and authenticating frames using time-covert channels may help in this regard, without requiring additional authentication frames.

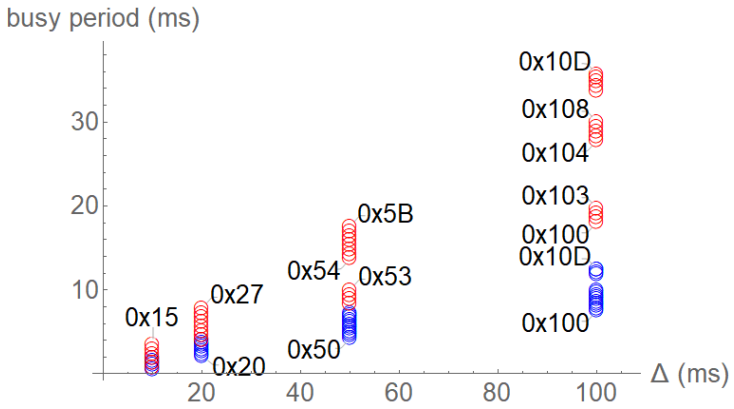


Figure 4.6: Busy period as computed for the 40 IDs in the setup (blue) and impact of doubling the number of IDs (red)

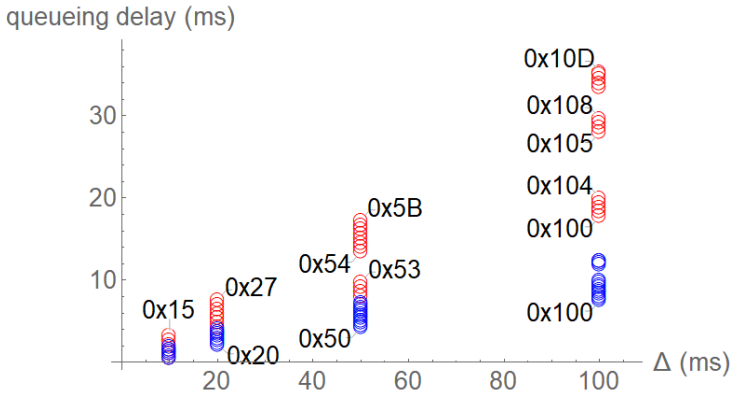


Figure 4.7: Worst-case queueing delay as computed for the 40 IDs in the setup (blue) and impact of doubling the number of IDs (red)

4.2 Related works

Even though previous research works [102], [103], [104] have studied the utilization of time covert channels in computer networks, there was less research done for time covert channels in Controller Area Networks [62], [96], [105]. With regards to optimization of traffic allocation, there are a few research works that have performed these studies in the past [106], [79], [107], [80] but without designing a time-covert channel after the traffic is optimized. Since the proposal from CANTO [30] is relying on frame arrival time for extracting the covert bits, there are related works which used these times for intrusion detection [108] and [109]. A different approach that uses Bloom filters [110] on frame arrival times together with the frame data to detect intrusions is proposed in [111].

4.3 Setup components

The embedded kit used to implement the frame optimization and time-covert authentication of frames is the AURIX TC224 Application Kit. This kit has an Infineon TC224 microcontroller that runs up to 133MHz core frequency and has 1MB of Flash memory and 96 kB of RAM memory. The CANoe tool from the PC together with the VN1640 adapter from Vector are used to locally record the traffic data. The local files are analyzed using the Mathematica tool, with respect to timing for frame scheduling optimization and time-covert authentication. All hardware components used in the experiments are shown in Figure 4.8.

The software application allows transmission of frames inside δ intervals using the CCU6 (Capture/Compare Unit) timer configured to trigger interrupts every $1\mu s$ as base configuration. The authentication data is computed in the main function using a MAC that takes into account the frame data and a frame counter, also incremented as part of the interrupt after a frame is transmitted. The last 7 bits of the MAC are used as ticks to introduce an additional delay to the base configuration of the timer interrupt. This delay is added to the cycle time of the frame together with the value of ϵ, also represented as ticks. Whenever the timer ticks have passed, the authentication delay ξ is set for the next timer interrupt. Whenever the interrupt is triggered, the frame is transmitted by the TC224 node. The configured values for ϵ and the cycle times for each frame are defined in the *MultiCAN+* software component. After initialization of the CAN driver with the 500Kbps data rate and initialization of the hardware timer, frames are transmitted with delays based on the value of ϵ and the authentication delay ξ from timer interrupts. After each frame is successfully transmitted, the frame counter is incremented and a new MAC value is computed for each frame using the counter and the data bytes.

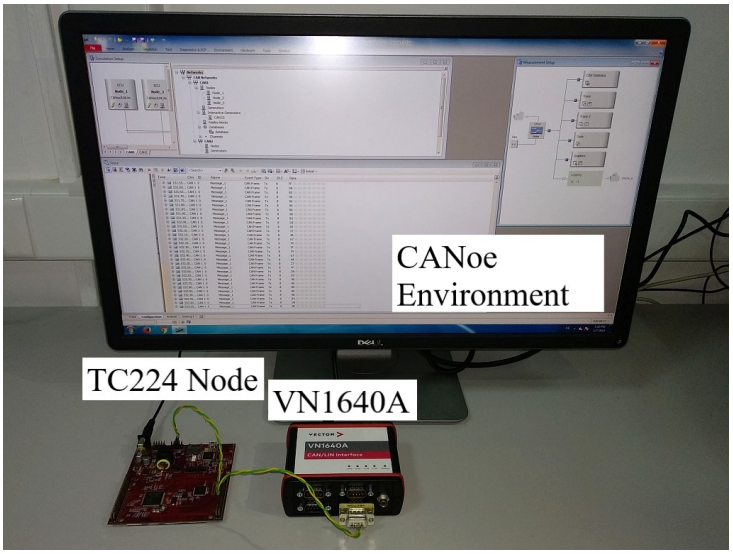


Figure 4.8: Experimental setup used for generating and recording CAN traffic according to the proposed mechanisms

4.4 Optimizing traffic allocation

Frame scheduling optimization is required for preserving the data rate of the time-covert channels. Four frame scheduling optimization algorithms are presented, providing the theoretical models and results for their implementation on the embedded device from the experimental setup. Then, the evaluation of the time-covert channel performance is done using two of these algorithms.

For each of the algorithms, several pairs of frame identifiers are defined with their cycle time represented as $\{(id_1, \Delta_1), (id_2, \Delta_2)$,..., $(id_n, \Delta_n)\}$. On-event frames are not considered since the focus is the design of time-covert channels for periodic messages. The timestamp when each frame is transmitted on the bus has to be measured and used by all receivers. In order to store the arrival time for each frame, pairs of frame identifiers and their reception time are considered as $\{(id_i, T_1^i), (id_i, T_2^i), ..., (id_i, T_l^i)\}$. Here, the timestamp for reception of frame id_i is $T_j^i, \forall j = 1..l$. An assumption one can have is that the arrival time difference between consecutive frames $T_{j+1}^i - T_j^i$ is equal to $\Delta_i, \forall i = 1..n, j = 1..l$. This is not the case, since the arrival time seen by a receiver node would be either faster or slower depending on the clock drift of the receiver relative to the transmitter. Arbitration between frames allows those with lower-value identifiers to be transmitted first whenever two nodes start a frame transmission at the same time. Considering that frames are delayed on the CAN bus due to arbitration, the difference between arrival times for consecutive cyclic frames increases even more, leading to a higher $T_{j+1}^i - T_j^i$ compared to $\Delta_i, \forall i = 1..n, j = 1..l$.

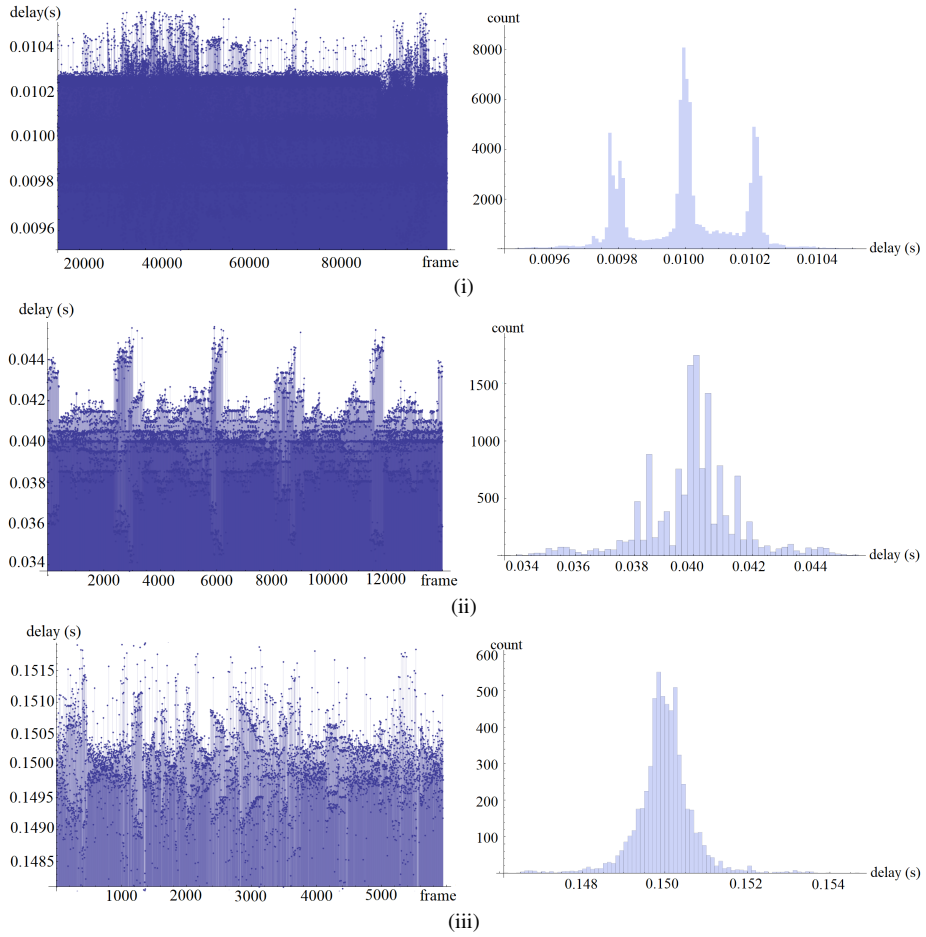


Figure 4.9: Frame arrival time and histogram distribution of frame arrival time, for frames arriving at (i) 10ms, (ii) 40ms and (iii) 150ms delays

Frame arrival time in real-world traces. In Figures 4.9 (i), (ii) and (iii), the arrival time and histograms for consecutive frames with a cycle time of $10ms$, $40ms$ and $150ms$ are shown. The delays from the expected frame arrival time are presented on the left side, while the number of frames that deviate from their nominal cycle time is depicted on the right side. As shown in these figures, it is common for frames with $10ms$ cycle time to deviate with up to $400\mu s$ while frames with $150ms$ cycle time may arrive $2 - 4ms$ later than their expected time. The time-covert channel requires frames to be delayed only by up to a few dozen microseconds from their expected arrival time in order to be considered authentic. In practice, frames are delayed by up to $4ms$ depending on their priority, so it is required to optimize frame scheduling to ensure the time-covert channel effectiveness.

4.4.1 Frame scheduling optimization

In case a node transmits two frames with specific cycle times on the bus, if the cycle times for the frames are $\Delta_i, \Delta_j, i, j = 1..n$, the frame transmission for them happens at the same time at multiples of $lcm(\Delta_i, \Delta_j)$. The lcm represents the least common multiple of the cycle times for frames i and j. For example, at every $150ms$, if $\Delta_i = 50ms$ and $\Delta_j = 75ms$, the transmission time for frames i and j is the same, but one of the frames is transmitted with a delay on the bus. This rule can be extended to any number of frames because their transmission is started at the same time on the bus at every multiple of $lcm(\Delta_1, \Delta_2, ..., \Delta_n)$, for n frames. Due to frame arbitration, any collision leads to one or more frames being transmitted with a delay from their expected cycle time. This affects the performance of the time covert channel because the delay causes arrival time to be out of the tolerance range. In order to prevent arbitration between cyclic frames from happening on the communication channel, a small delay is considered for each frame. An extension to all frame identifiers and their cycle times with the additional delay is $\{(id_1, \Delta_1, \epsilon_1), (id_2, \Delta_2, \epsilon_2), ..., (id_n, \Delta_n, \epsilon_n)\}$ where $\epsilon_i, i = 1..n$ is the delay. This leads to a transmission and arrival time for frames at multiples of their cycle time with a small offset, i.e., $k\Delta_i + \epsilon_i$, compared to only using their cycle time, i.e., $k\Delta_i$. The intention is to find a set of offsets, $\epsilon_i, i = 1..n$, that ensures there are no collisions between legitimate frames that implement the time-covert channel, while maximizing the inter-frame space, whenever possible.

In order to define the frame scheduling optimization algorithms, the dataset shown in the relation below is used. The dataset from Equation 4.3 contains the information for n frame identifiers together with their cycle time and the offsets that are set for each frame.

$$\begin{cases} T_1 = \{(id_1, \epsilon_1), (id_1, \Delta_1 + \epsilon_1), ..., (id_1, (l-1)\Delta_1 + \epsilon_1)\} \\ T_2 = \{(id_2, \epsilon_2), (id_2, \Delta_2 + \epsilon_2), ..., (id_2, (l-1)\Delta_2 + \epsilon_2)\} \\ \\ \qquad\qquad\qquad\qquad ..... \\ T_n = \{(id_n, \epsilon_n), (id_n, \Delta_n + \epsilon_n), ..., (id_n, (l-1)\Delta_n + \epsilon_n)\} \end{cases} \tag{4.3}$$

The frame scheduling is considered to be optimal only if $|T_1 \cup T_2 \cup ... \cup T_n| = |T_1| + |T_2| + ... + |T_n|$ where $|T_i|, i = 1..n$ is the cardinality for every set from the previous relation. The expectation is that all frames arrive at the expected time based on the scheduling table. Considering that $T^* = \{t_1, t_2, ..., t_n\}$ is a set of the recorded timestamps for k frames transmitted on the bus, the inter-frame time between each two frames is $t_i - t_{i-1}, i = 1..k$. In order to maximize the inter-frame time, a quality factor, q, that is used needs to be minimized. The quality factor, q, is defined as:

$$q = \frac{1}{n} \sum_{i=2}^{n} \frac{1}{t_i - t_{i-1}} \tag{4.4}$$

The quality factor from Equation 4.4 needs to have a low value in order to minimize

the sum of inter-frame spaces and optimize the frame scheduling algorithms. Since the time between consecutive frames, $t_i - t_{i-1}, \forall i = 2..n$, is represented in seconds, the quality factor is measured in Hz or s^{-1}.

A practical instantiation. As inputs defined for optimization of the frame scheduling in the algorithms that follow, the array of frames that is used with the required cycle time specified in milliseconds is $\overline{\Delta}$:

$$\overline{\Delta} = \{\underbrace{10, 10, ..., 10}_{\times 6}, \underbrace{20, 20, ..., 20}_{\times 8}, \underbrace{50, 50, ..., 50}_{\times 12}, \underbrace{100, 100, ..., 100}_{\times 14}\} \tag{4.5}$$

The array from Equation 4.5 contains 6 frames with a cycle time of $10ms$, 8 frames with the cycle time of $20ms$, 12 frames with the cycle time of $50ms$ and 14 frames with the cycle time of $100ms$. This means a total of 40 distinct frames with various cycle times for which $\epsilon_i, i = 1, n$ is added in order to delay the transmission time for each iteration, based on the predefined scheduling table.

Binary Symmetric Allocation. This is the first algorithm that is defined, which is both easy to use and provides the expected results with regards to timings of frames transmitted on the bus. The window size, w, defined for the scheduling optimization algorithm, is the minimum cycle time value. The offset ϵ_1 is set at half of the window size while the following offsets are set at one quarter and three quarters of the window size, w. The same logic is applied to the following steps until all frames are allocated in the scheduling table. The expected frame timings with a detailed overview of the delays lower than $500\mu s$ or $1.2ms$ are shown on the left and right side of Figure 4.10. The experimental results presented in Figure 4.11, as graphical representation and histogram, show a clear relation between theory and implementation but with some differences in the maximum values for inter-frame time. The delays are expected to reach the value of $1.2ms$, but based on the experimental results, there are some delays between frames of $2.5ms$. These are caused by the transmitters, in case they miss the transmission time slot, and the frame is sent at a later point in time. For the Binary Symmetric Allocation, $150\mu s$ is the minimum inter-frame time, which seems to be too short for some frames.

Randomized Search Allocation. This is the second algorithm that is defined, based on a set of offsets, $\epsilon_i, i = 1..n$, which are equally spaced inside the defined time window, w, for the frame scheduling optimization. The inter-frame space is computed to be the minimum cycle time of the frames divided by the count, i.e., $\min(\Delta_1, \Delta_2, ..., \Delta_n)/n$. Other values for the inter-frame space also exist in the algorithm design as the theoretical model and experimental results show. All offsets are randomly allocated for each frame for a preset number of iterations. An improvement of the resulted offset list is possible by increasing the number of iterations. The results obtained using the Randomized Search Allocation algorithm for frame scheduling are better than those for the Binary Symmetric algorithm, but there are still some delays higher than the theoretically expected ones,

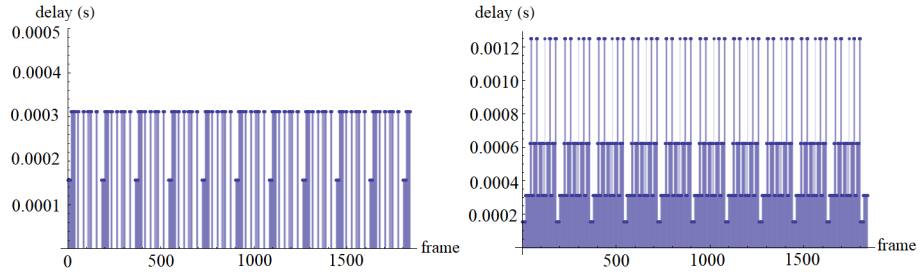


Figure 4.10: Theoretical displacement of delays in case of Binary Symmetric Allocation: detail for delays lower than $300\mu s$ and overall view up to $1.2ms$ for the first 2,000 frames

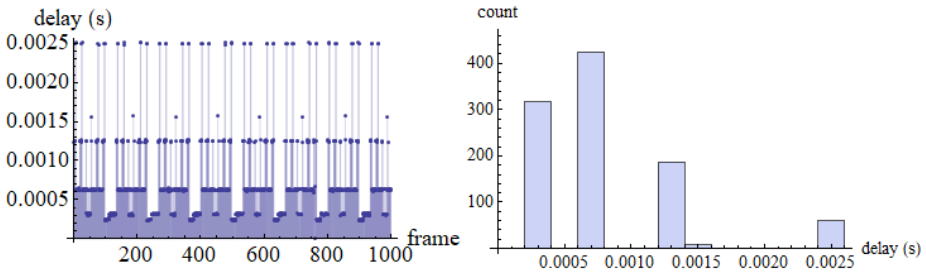


Figure 4.11: Experimental measurements from CANoe of an Infineon node broadcasting after Binary Symmetric Allocation: delays (left) and histogram distribution of delays (right)

when running the experiments.The theoretical distribution of delays is shown in Figure 4.12, while the experimental results are presented in Figure 4.13. There are some inter-frame times of $1.5ms$ or $2.5ms$ even though the maximum expected delays are of $1.2ms$. This happens for the same reason as for the Binary Symmetric Allocation, due to computational delays for the transmitter that misses the transmission time slot. This means that a minimum value of $250\mu s$ between frames is still too short, so an extension of the minimum inter-frame space is done for the algorithms that are described in what follows.

Greedy Allocation. This is the third algorithm that is defined. It is based on a set of offsets, $\epsilon_i, i = 1..n$, that are generated similarly to the Randomized Search Allocation algorithm. In order to optimize the delay offset value distribution, so, to minimize q, the allocation of delays is performed in ascending order based on the frame cycle time. The timing issue identified for the first two algorithms is also applicable for the Greedy Allocation as well since the minimum inter-frame time is also set to $250\mu s$. In order to mitigate this limitation, an update for the Greedy Allocation with a multi-layer version is defined, and explained in what follows.

Multi-Layer Greedy Allocation. This is the extension of the Greedy Allocation algorithm called Multi-Layer Greedy Allocation. Compared to it, in order to increase the minimum inter-frame space time, the way the offsets, i.e., $\epsilon_i, i = 1..n$, are generated is changed. Now, instead of selecting the delays based on e as before, delays

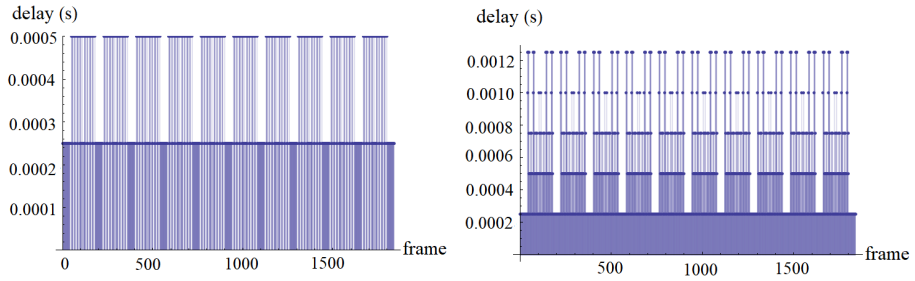


Figure 4.12: Theoretical displacement of delays in case of Randomized Search Allocation: detail for delays lower than $500\mu s$ and overall view up to $1.2ms$ for the first 2,000 frames

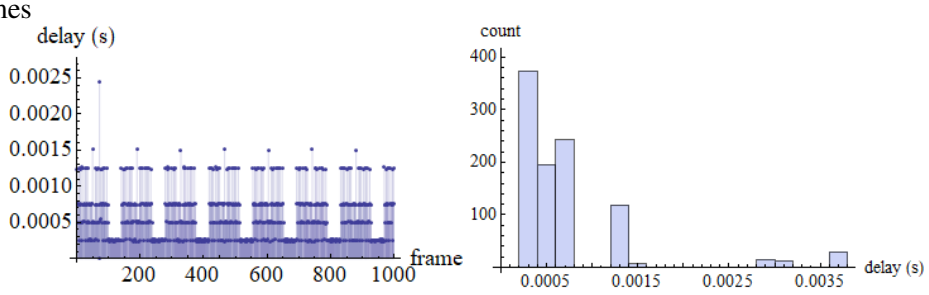


Figure 4.13: Experimental measurements from CANoe of an Infineon node broadcasting after Randomized Search Allocation: delays (left) and histogram distribution of delays (right)

are selected based on any multiple of e that is lower than the cycle time of the frame Δ_i. This means that frames with larger cycle times have a higher offset outside of the ranges defined for Randomized Search or Greedy Allocation algorithms which are of 0 to $\min(\Delta_i), i = 1..n$. The theoretical distribution of delays for Multi-Layer Greedy Allocation is shown in Figure 4.14 while the experimental results are presented in Figure 4.15. For the Multi-Layer version of Greedy Allocation, the minimum inter-frame space is extended to $500\mu s$ which is double compared to $250\mu s$ from the normal Greedy Allocation. This allows the algorithm to provide experimental results with the same patterns as the theoretical models, without additional delays between frames caused by nodes missing frame transmission time slots due to a low inter-frame space.

GCD-based Allocation. This is the last optimization algorithm that is defined, also named Circular-GCD algorithm. It is based on the allocation of delays with at least $500\mu s$ inter-frame space and a small offset, $\delta_i, i = 1..n$. The small offset is a multiple of the $\gcd(\Delta_1, \Delta_2, ..., \Delta_n)$ with the condition that $500\mu s + \delta_i$ does not extend the cycle time defined for the frame, Δ_i. As experimental values, the array was configured as shown in Equation 4.6:

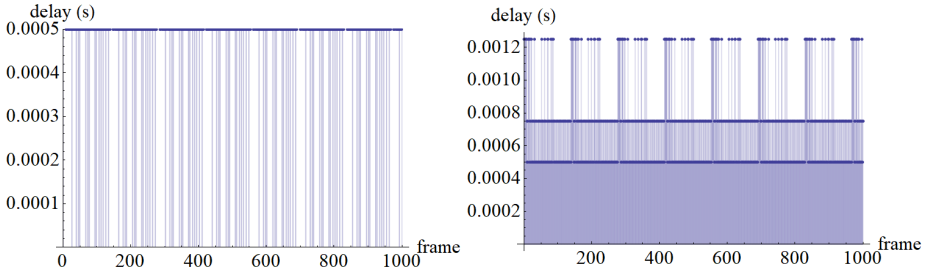


Figure 4.14: Theoretical displacement of delays in case of Multi-Layer Greedy Allocation: detail for delays lower than $300\mu s$ and overall view up to $1.2ms$ for the first 2,000 frames

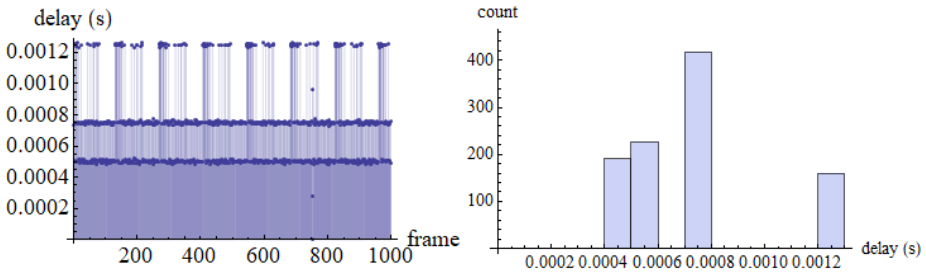


Figure 4.15: Experimental measurements from CANoe of an Infineon node broadcasting after Multi-Layer Greedy Allocation: delays (left) and histogram distribution of delays (right)

$$\begin{aligned}\epsilon_{i:1,40} = \{&0, 0.5, 1.0, 1.5, 2.0, 2.5, 3.0, 13.0, 3.5, 13.5, 4.0, 14.0,\\ &4.5, 14.5,\ 5.0, 15.0, 25.0, 35.0, 45.0, 5.5, 15.5, 25.5, 35.5,\\ &45.5, 6.0, 16.0, 26.0, 36.0, 46.0, 76.0, 86.0, 96.0, 6.5, 16.5,\\ &26.5, 36.5, 46.5, 56.5, 66.5, 76.5\}\end{aligned} \tag{4.6}$$

The theoretical distribution of delays for the GCD-based Allocation is shown in Figure 4.16, while the experimental results are presented in Figure 4.17. For the GCD-based Allocation algorithm, the minimum inter-frame space is extended to $600\mu s$, but this limits the maximum inter-frame space to $2.2ms$. Similar to the Multi-Layer greedy, GCD-based Allocation works as expected for the experiments, so the experimental results show the same patterns as the theoretical models. By limiting the minimum inter-frame space to $500\mu s$, the maximum inter-frame space is of $4ms$. Since the design goal is to minimize the quality factor, q, without maximizing the inter-frame space, the values of $500\mu s$ and $4ms$ are used for the minimum and maximum IFS when computing the value of q.

A summary of the frame scheduling optimization algorithms that shows the quality factor, minimum and maximum inter-frame spaces is presented in Table 4.1. The minimum inter-frame space from the Binary Symmetric search, Randomized Search and

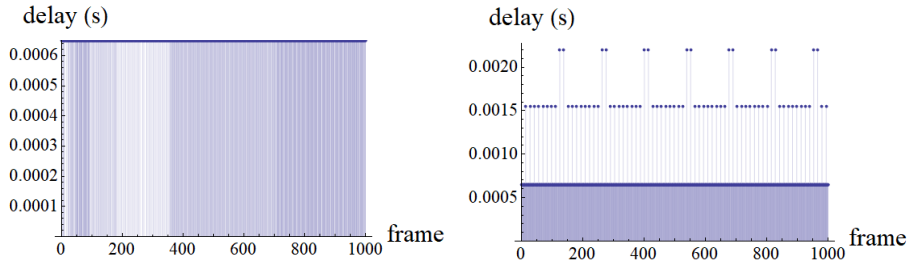


Figure 4.16: Theoretical displacement of delays in case of GCD Allocation at $0.6ms$ allocation: detail for delays lower than $300\mu s$ and overal view up to $1.2ms$ for the first 2,000 frames

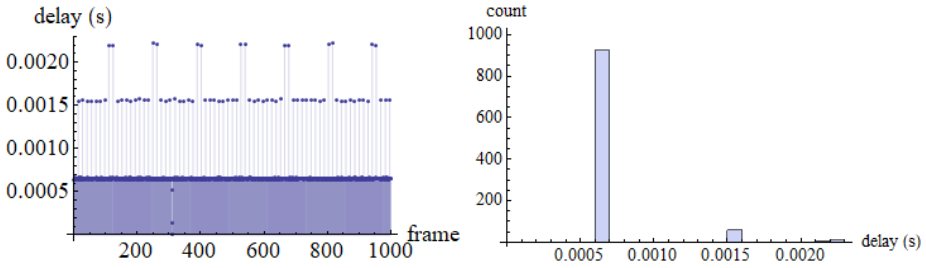


Figure 4.17: Experimental measurements from CANoe of an Infineon node broadcasting after GCD Allocation at $0.6ms$ inter-frame space: delays (left) and histogram distribution of delays (right)

Greedy algorithms turns out to be problematic since it is lower than or close to the time it takes a node to transmit a frame on the vehicle bus. Whenever the scheduled minimum inter-frame space is of $150\mu s - 250\mu s$, in case of arbitration, each frame that wins the bus delays the transmission time of the next one, in a cascading effect. Then, when the bus reaches idle state for some time, the frame transmission is restarted according to the scheduling table. Unfortunately, the minimum inter-frame space of $150\mu s - 250\mu s$ is not enough for the time covert channel to work without any performance degradation. On the other hand, due to lack of collisions between frames, the Multi-Layer Greedy and Circular GCD Allocation algorithms can be used for implementing a time-covert chan-

Table 4.1: Comparison of allocation algorithms

	Completeness	q-Factor (s^{-1})	Min IFS (ms)	Max IFS (ms)
Binary Sym. Search	✓	2.37	0.15	2.5
Randomized Search	✓	2.51	0.25	3.75
Greedy Search	✓	2.39	0.25	2.5
Greedy ML Search	✓	1.50	0.5	1.25
Circular GCD	✓	1.86	0.5	4

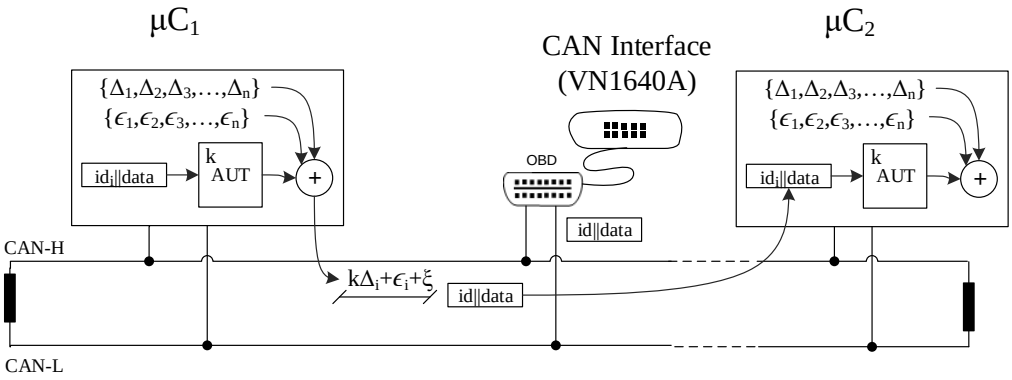


Figure 4.18: Basic depiction of the addressed scenario: ECUs sending/receiving packets on the CAN bus, authentication data is encoded in delays

nel. Additionally, the busload is uniform whenever the values of minimum and maximum inter-frame space are close. This can be seen as an option for practical implementations. Now, the algorithms are classified based on the quality factor in ascending order. The Multi-Layer Greedy provides the best allocation with $q = 1.50$, followed by Circular GCD with $q = 1.86$, Binary Symmetric Allocation with $q = 2.37$, Greedy search with $q = 2.39$ and Randomized Search Allocation with $q = 2.51$.

4.4.2 A time-covert authentication protocol

The design is based on a covert timing channel that provides authentication of frames without using any byte from the data field. A summary of the proposal is shown in Figure 4.18, where 2 nodes communicate on the CAN bus. Each node transmits frames that have specific cycle times, $\Delta_i, i = 1..n$, with a small delay ϵ_i and and offset computed as authentication data ξ. The offset is based on the frame identifier and computed over the data bytes. The receiving node computes the offset, ξ, using the frame data field and then verifies that each frame arrived with the expected drift ξ from its cycle time, taking into account the small delay ϵ_i. The traffic is logged on a PC using a CAN Interface connected to the same bus, e.g. a VN1640A from Vector, in order to evaluate the timings for each frame sent on the network. Some of the improvements that timing covert channel with authenticated delays for CAN communication bring are:

- Frame authentication does not require additional bytes from CAN frames,
- Additional frames with authentication data to be transmitted on the bus are not required,
- It preserves the busload since authentication data is based only on the frame transmission timing.

The design goal for the frame scheduling optimization algorithms is to allow the time covert channel from the previous work of the author, INCANTA [24], to work without having its performance affected by frame collisions due to arbitration. The same implementation of the time covert channel is considered as it was done in INCANTA [24], with small changes because now all frames also have the additional offset $\epsilon_i, i = 1..n$ that results from the frame scheduling optimization algorithms. Thereby, the protocol used for frame authentication using the time covert channel is denoted as INCANTA and described below, taking into account its name from the previous research paper [24]. Each node that is part of the time covert channel has to follow the steps shown below:

1. $\mathsf{SendCyclic}(id_i, \mathrm{m})$ where each node transmits a frame with the identifier id_i and the message content m at a fixed time interval $k\Delta_i + \epsilon_i$. The value of Δ_i is the cycle time of the frame, ϵ_i is defined as part of the frame scheduling optimization algorithm and the authentication value, denoted as ξ or tag, is computed before each transmission by the node as $MAC_{\mathsf{k}}(k, id_i, \mathrm{m}), i = 1..n$, using the iteration for individual frames k_i as freshness counter and the content of the message, m, as input for the MAC. After $k\Delta_i + \epsilon_i$ time has passed, the sender waits for tag timer ticks before transmitting the frame.
2. $\mathsf{RecCyclic}(id_i, \mathrm{m})$ where each node verifies if the received message falls within the expected tolerance of the time covert authentication protocol based on the received time. This means that for each iteration, k, nodes that receive the frame compute $MAC_{\mathsf{k}}(k, id_i, \mathrm{m}), i = 1..n$ and verify if $|\mathsf{t}_k - \mathsf{t}_{k-1}| - (\Delta_i + \mathsf{T}_k - \mathsf{T}_{k-1})| \leq \rho$ to ensure the frame was received at time $k\Delta_i + \epsilon_i + \xi$, within the tolerance ρ. If the arrival time is within expected tolerance, ρ, the frame is considered legitimate. Otherwise, if it is received outside of the expected range, it is considered as an intrusion.

A secret key is shared between nodes, k, used in the protocol for computing the message authentication code $MAC_{\mathsf{k}}(k, id_i, \mathrm{m}), i = 1..n$. No key exchange is required to be performed between nodes. The assumption is that the key is known by the nodes before the time-covert channel is established. The scheduling table, T^*, shown in Equation 4.7 is also pre-shared by the nodes.

$$T^* = \{(id_1, \Delta_1, \epsilon_1), (id_2, \Delta_2, \epsilon_2), ..., (id_n, \Delta_n, \epsilon_n)\} \tag{4.7}$$

This is required, so genuine receivers are able to compute the expected time for each frame upon arrival using the expected cycle time Δ_i and the offset from the scheduling optimization protocol ϵ_i for each frame id_i. During the experiments, the accumulated clock skews for different nodes are also seen as a contributor for the measured delays, since they affect the receive time even more than stuffing bits, relative to the expected arrival time for each frame.

4.4.3 Adversary model

The adversary model considered for the security analysis is the Dolev-Yao adversary [86] with full control over the CAN bus. Another assumption in the design of the time-covert channel is that genuine nodes use pre-shared keys and frame scheduling tables, while adversarial nodes are external and do not have this knowledge. This is a realistic approach since the car manufacturers are able to set keys inside the vehicle units or to request the suppliers to do so without making them public. Usually, attacks on in-vehicle buses are external and do not have inside information regarding previously shared knowledge between nodes. In case a genuine transmitter is replaced as a result of an attack, the adversarial node needs to transmit all the IDs expected from the genuine node following the scheduling table and using the message authentication code. This means the adversary success rate can be defined as the ratio between the delay tolerance, ρ, and the protocol security level ℓ, as shown in Equation 4.8:

$$\gamma_{adv} = \frac{\rho}{2^{\ell}} \tag{4.8}$$

The experiments show that a value for the delay tolerance, ρ, of up to $5\mu s$ can be used. Considering that standards which define time-synchronization methods [98] require an accuracy error of maximum $10\mu s$ between CAN nodes, an extension up to $10\mu s$ for the tolerance used by the proposed time-covert channel can be considered.

4.4.4 Results with optimized traffic and a single sender

Since the CAN baud rate configuration is of 500Kbps, the worst-case frame duration is of $270\mu s$. This means that, by using the Multi-Layer Greedy or Circular-GCD Allocation algorithms which allow a minimum IFS of $500\mu s$, the bus is idle for at least $230\mu s$. The security level that can be achieved for one frame is of 8 bits, as truncated MAC, which results in an authenticated delay of up to $255\mu s$. Even though this will cause collisions on the communication channel depending on the frame duration, reducing it to 7 bits, as truncated MAC, can mitigate the risk and allow authentication delays to be done with at most $127\mu s$. The idle bus time is the main reason for choosing Greedy-ML and Circular-GCD as frame scheduling algorithms. Due to frames that may vary in length due to the number of stuffing bits, not all frames are deemed authentic, even for the 7 bit truncated MAC. The inter-frame space available on the communication channel is shown in Figure 4.19, without (i) and with (ii) the covert channel in place while using Circular GCD optimization algorithm. As expected, without the covert channel in place, there is no deviation seen for the frame scheduling algorithm that allows a minimum IFS of $500\mu s$. Comparing the deviations on the right side of Figure 4.19, it is obvious that if there is no time covert channel in place, the delays are always close to the expected value. When the time covert channel is used, the variations are random based on the value of the 7-bit truncated MAC with deviations that may be of up to $127\mu s$. The variations are

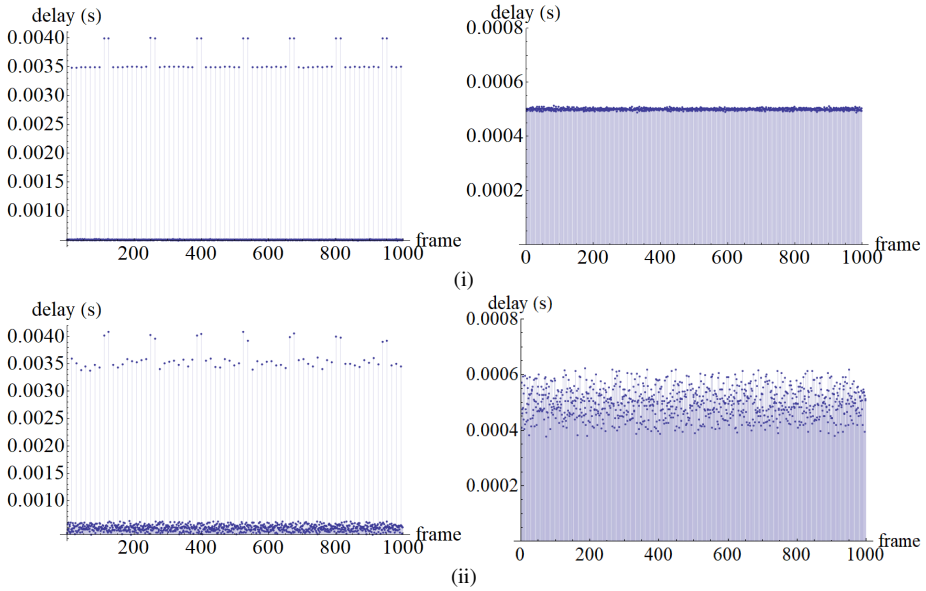


Figure 4.19: Interframe delays (broadcast from Infineon TriCore node) in case of Circular GCD optimization $\epsilon = 0.5$ without a covert channel (i) and with the covert channel in place (ii)

unpredictable for an adversary that cannot compute the value of the MAC since it does not have knowledge of the secret key k so the time covert channel is resilient to replay attacks.

All of the experiments show a good distribution of reception time, with a deviation of up to $\pm 15\mu s$ from the expected time. This variation may be influenced by a different number of stuffing bits in consecutive frames since $15\mu s$ is the time it takes to transmit 7-8 bits at a bit rate of 500Kbps for CAN. Additional or fewer stuffing bits influence the frame length. Without taking into account this change, the actual reception time differs from the expected one. Deviations from the anticipated time for IDs transmitted at $10ms$ (i), $20ms$ (ii), $50ms$ (iii) and $100ms$ (iv) are shown on the left side in Figure 4.20. On the right side, the deviation time histogram distribution is presented for the same IDs. This shows that most of the deviations are inside the $\pm 5\mu s$ interval with few values that go up to $\pm 10\mu s$ or $\pm 15\mu s$ for the ID with a $10ms$ cycle time. The variation is reduced to $\pm 5\mu s$ from the expected reception time if stuffing bits from each frame are counted and taken into account for the frame duration, as presented in what follows.

In Figure 4.21, the experimental data is shown for the same IDs that are visually depicted in Figure 4.20. On the left side of Figure 4.21, the deviations shown are in the $\pm 5\mu s$ range. On the right side of Figure 4.21, in the histogram distribution, most of the values that are shown are in the $\pm 2\mu s$ range. This means that the deviation represents only a 1 bit time difference for the 500Kbps CAN bit rate. Since the overall deviation is in

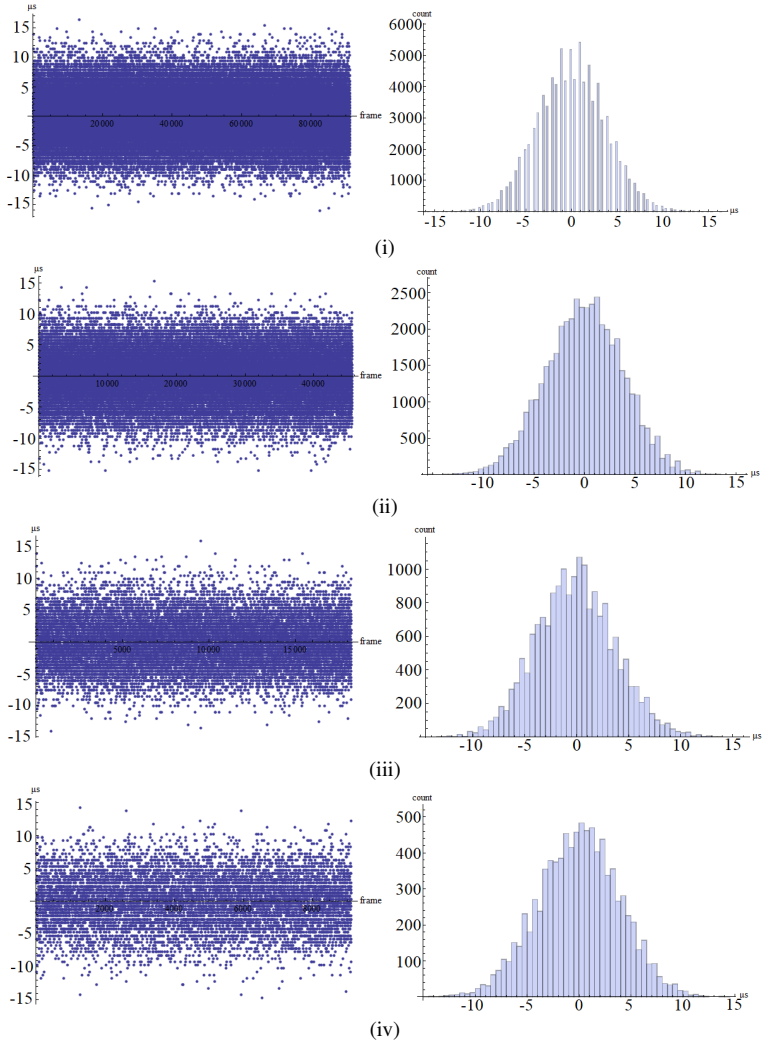


Figure 4.20: Experimental measurements for covert authenticated frames from an Infineon TriCore node: deviation from the expected delays (left) and histogram distribution (right) for an ID sent at $10ms$ (i), $20ms$ (ii), $50ms$ (iii) and $100ms$ (iv)

the $\pm 5\mu s$ range, by setting the tolerance ρ to $5\mu s$, it would lead to a 100% success rate for frames transmitted on the time-covert channel to be seen as legitimate. Setting different values for the tolerance ρ would lead to various success rates for genuine transmitters and adversarial nodes based on the time a frame is received if compared to the expected reception time. This is shown in Table 4.2 where single frame success rates vary from 93.34% to 100% for genuine transmitters and 3% to 7.8% for adversarial nodes when the tolerance ρ is set in the $2 - 5\mu s$ range. The results are based on the experiments

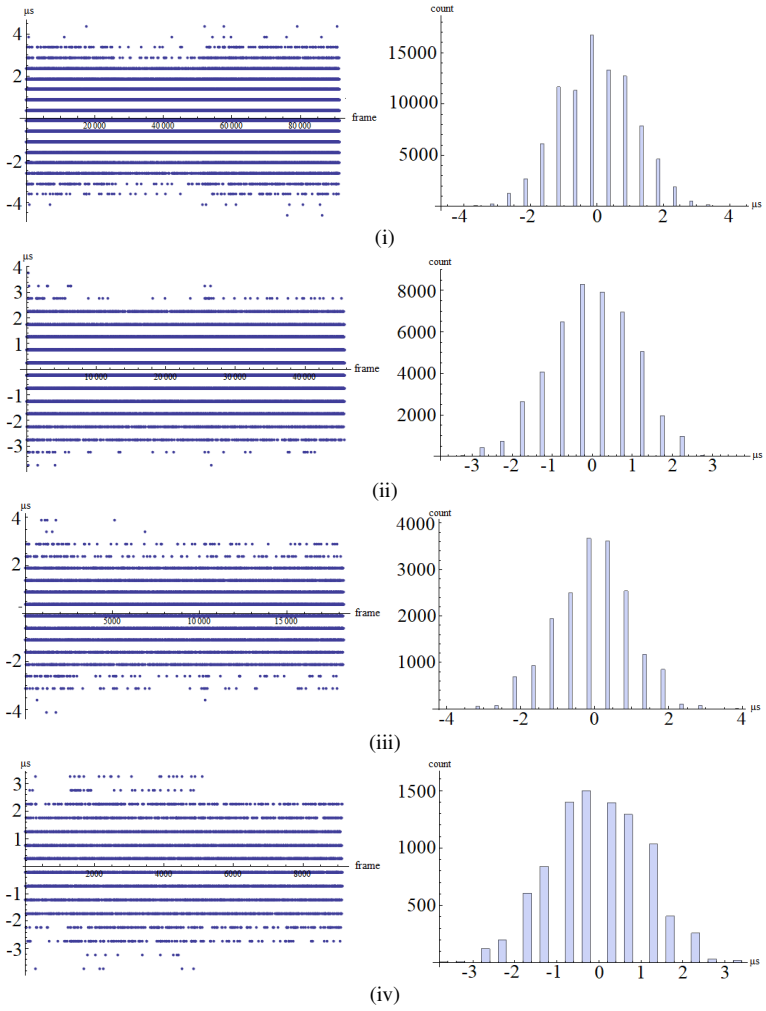


Figure 4.21: Experimental measurements for covert authenticated frames from an Infineon TriCore node: deviation from the expected delays (left) and histogram distribution (right) for an ID sent at $10ms$ (i), $20ms$ (ii), $50ms$ (iii) and $100ms$ (iv)

Table 4.2: Success rates (%) with tolerance $\rho \in \{2, 3, 4, 5\}\mu s$ at $\ell = 7$

ρ		$k=1$	$k=2$	$k=3$	$k=4$	$k=6$
$2\mu s$	γ_{ecu}	93.34	87.14	81.34	75.93	66.10
	γ_{adv}	3.1	0.09	0.003	0.00009	9.3×10^{-8}
$3\mu s$	γ_{ecu}	99.56	99.12	98.68	98.25	97.38
	γ_{adv}	4.7	0.22	0.01	0.0004	1.1×10^{-6}
$4\mu s$	γ_{ecu}	99.99	99.98	99.97	99.96	99.94
	γ_{adv}	6.2	0.39	0.02	0.001	5.9×10^{-6}
$5\mu s$	γ_{ecu}	100	100	100	100	100
	γ_{adv}	7.8	0.62	0.04	0.003	0.00002

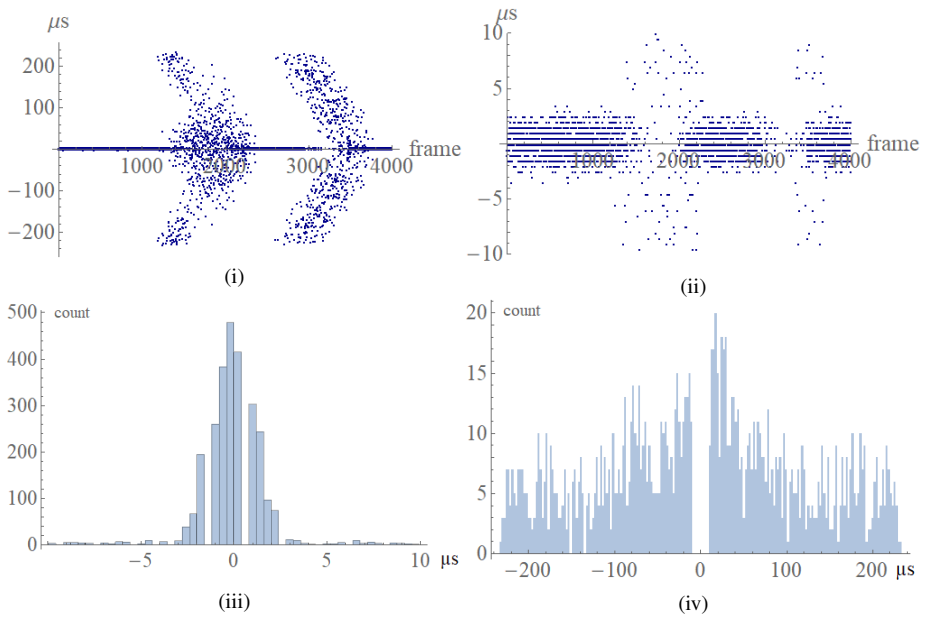


Figure 4.22: Gaps due to synchronization loss in the expected arrival time of an ID (i), detailed view in the $[-10\mu s, 10\mu s]$ range (ii), histogram distribution of deviations in the $[-10\mu s, 10\mu s]$ range (iii) and histogram distribution of deviations in the $[-200\mu s, -10\mu s] \cup [10\mu s, 200\mu s]$ range (iv)

performed on 1.2 million frames. Acceptance rate for genuine transmitters was computed as mean value of acceptance rates for all frames. The values for the acceptance rates are of 91.4% to 95.2% for the $2\mu s$ tolerance, 99.21% to 99.93% for the $3\mu s$ tolerance and 99.96% to 100% for the $4\mu s$ tolerance. The acceptance rates for adversarial transmitters are computed with the tolerance set to $10\ mus$ for the $\pm 5\mu s$ range and the security level ℓ set to 7 bits. In order to improve the time-covert channel against adversarial transmitters, the acceptance rate is defined in Equation 4.9 over multiple frames, as formalized in the following Equation, where k is the number of frames:

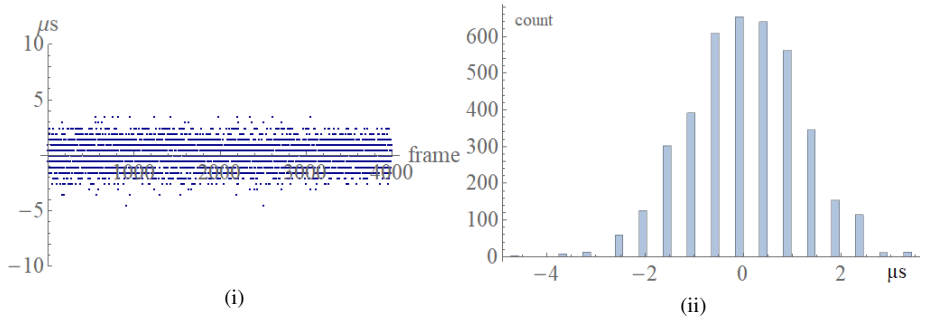
$$\gamma_{\blacklozenge}(k) = \gamma_{\blacklozenge}^{k}, \blacklozenge \in \{adv, ecu\} \tag{4.9}$$

As also shown in Table 4.2, by using the acceptance rate over 6 frames for a tolerance set to $5\mu s$, all frames transmitted by genuine nodes are seen as legitimate. The false positive rate is of 2 out of 10 million frames, i.e., frames transmitted by adversarial nodes that are seen as legitimate.

4.4.5 The multi-sender case and noisy channels

In order to evaluate the time covert channel in realistic scenarios, the performance for a scenario with multiple senders and the influence of un-optimized traffic are analyzed. In case of multiple senders, the observed issues are related to desynchronization between nodes and clock skew accumulation. These problems are solved by using periodic re-synchronization and de-skewing of the clocks. Additionally, frames from un-optimized traffic collide with the frames that are transmitted based on the scheduling tables and cause delays that decrease the performance of the time-covert channel.

In order to define a multi-sender setup, the implementation of the time covert channel is ported to a second node, an Infineon AURIX TC237 microcontroller, using a similar application kit as for the Infineon AURIX TC224 node. Both nodes communicate on the CAN bus together with the Vector VN1640 equipment used to log the transmitted frames. The time covert communication is started by the nodes after receival of a start frame sent by the VN1640 from the CANoe tool. Loss of synchronization between nodes is noticed while analyzing the collected logs for a frame with a $10ms$ cycle time that is shown in Figure 4.22 (i). Delays from the expected time are within the range of $\pm 200\mu s$ which is close to the frame duration of a 64-bit frame on CAN for a bit rate of 500Kbps. This is the blocking delay B_m that was earlier described for the worst-case arrival time on the CAN bus. The analyzed areas from Figure 4.22 are split in the ranges of $\pm 200\mu s$ (i) and $\pm 10\mu s$ (ii) showing the histograms for both areas in (iii) and (iv). Based on the experimental results, it is clear that most of the values are within the $\pm 10\mu s$ range. In order to improve the results, based on the measured clock skew of one node relative to the other, an update is done in the software implementation to reduce the accumulated clock skew effect. The TC237 code is updated such that the current time of TC237 is taking into account the skew of 0.999965645, relative to TC224's clock. This means that the node updates its current internal time with $3.4355\mu s$ every $100ms$, $citime = citime - 3.4355 * (citime/100,000)$. This update is required due to the skew accumulation, which means that every 1 second, the node needs to adjust its internal clock with $34.355\mu s$ because the measured skew difference is of $1s - 0.999965645s = 0.000034355s$, following the recommendation from existing works which study skew adjustment techniques [112]. The measured delays after performing the de-skewing of TC237 relative to TC224 are shown in Figure 4.23. Now all the values are within the $\pm 5\mu s$ range and in line with the previous experiments, with optimized traffic and a single sender, which means that node synchronization and clock de-skewing is successful. Even though the multiple sender scenario is verified using only two nodes, extending this to more than two is possible by following the same approach with regards to re-synchronization of transmitter clocks. This is also required as part of AUTOSAR time synchronization protocol for CAN [98], where the maximum allowed tolerance is of $10\mu s$ between all nodes. This synchronization error is used as tolerance value for the time covert channel, but it does not affect the frame scheduling optimization algorithms, which allow a higher inter-frame space of $150\mu s$ to $500\mu s$.



(i) (ii)

Figure 4.23: The same ID after de-skewing: deviations from the arrival time (i) and their histogram distribution (ii)

The un-optimized scenario is based on a log collected from a real-world vehicle that is replayed on the CAN bus using the Vector tool. There are two tests that are performed, one using half of the log data, i.e., 18% car traffic, and the other with the full log data, i.e., 36% car traffic. The TC224 contribution is of 18% of the busload in both scenarios while sending frames based on the frame scheduling algorithm and using the time-covert channel.

The results from both scenarios, multiple senders and single senders with un-optimized traffic from the car are shown in Table 4.3. The same values for the tolerance $\rho \in \{2, 3, 4, 5\}\mu s$ and for the security level $\ell = 7$ are used as in the previous experiments. This table contains information relative to multiple sender transmission and the impact of implementing the frame scheduling algorithm without periodic synchronization between nodes. It also contains the effect of re-synchronization and of de-skewing to the acceptance rates. The success rates for legitimate nodes are affected if synchronization or de-skewing is not performed. In case they are, the acceptance rate is of 99.68% for a $\pm 5\mu s$ tolerance, similar to the single-sender approach, and very close to 100%. This means that extension to multiple senders is possible if they are re-synchronized and the clock skew between them is periodically adjusted. The final rows from Table 4.3 show the results for the time covert channel for a single sender over un-optimized traffic. In case of 18% un-optimized traffic, the success rate is at most 65.81% for a tolerance of $\pm 5\mu s$, while for the same tolerance, in case of 36% un-optimized traffic, the success rate is 38.7%. In order to improve the performance of the time-covert channel, receivers must use k-out-of-n frames instead of single frames. This will also increase the chances for adversarial nodes to transmit frames that are considered legitimate. The success rate for both genuine and adversarial nodes is defined in Equation 4.10 based on the binominal distribution:

$$\gamma_{\blacklozenge}(k, n) = \sum_{l=k}^{n} \binom{n}{l} \gamma_{\blacklozenge}^{l} (1 - \gamma_{\blacklozenge})^{n-l}, \blacklozenge \in \{adv, ecu\} \tag{4.10}$$

Table 4.3: Success rates (%) for legitimate frames in different scenarios ($\rho \in \{2,3,4,5\}\mu s$ and $\ell = 7$)

Scenario	ρ			
	$2\mu s$	$3\mu s$	$4\mu s$	$5\mu s$
Single Sender	93.34	99.56	99.99	100
Dual Sender (opt.)	59.19	71.29	77.76	80.25
Dual Sender (opt./sync.)	69.38	76.40	77.56	77.89
Dual Sender (opt./sync./de-skew)	85.07	95.52	98.68	99.68
Single Sender (on 18% car traffic)	34.30	50.81	61.70	65.81
Single Sender (on 36% car traffic)	17.75	28.02	35.70	38.70

Table 4.4: Success rates (%) for k-out-of-n scheme with tolerance $\rho \in \{2,3,4,5\}\mu s$, $\ell = 7$, $n = 24$ over 18% car traffic

ρ		$k=6$	$k=8$	$k=10$	$k=12$	$k=14$
$2\mu s$	γ_{ecu}	88.2321	61.5313	28.7884	8.23784	1.35971
	γ_{adv}	0.00770	0.00004	1.2×10^{-7}	1.6×10^{-10}	1.2×10^{-13}
$3\mu s$	γ_{ecu}	99.7402	97.3359	86.4570	61.1860	29.7967
	γ_{adv}	0.06864	0.00087	5.4×10^{-6}	1.8×10^{-8}	3.1×10^{-11}
$4\mu s$	γ_{ecu}	99.9946	99.8746	98.5952	91.6052	71.2112
	γ_{adv}	0.30101	0.00689	0.00007	4.7×10^{-7}	1.5×10^{-9}
$5\mu s$	γ_{ecu}	99.9992	99.9727	99.5740	96.5006	83.8611
	γ_{adv}	0.89451	0.03255	0.00059	5.7×10^{-6}	2.9×10^{-8}

The values for k and n are chosen by following the experimental results. They are shown together with the acceptance rates in Table 4.4. By allowing an arrival time of 6 out of 24 frames to be within the expected tolerance, the success rate from Table 4.3 increases from 65.81% for $5\mu s$ tolerance to 99.9992% (which is very close to 100%). Increasing the value of k reduces the performance of the time-covert channel but also reduces chances for adversarial nodes to inject frames that are considered legitimate. The assumption of use for the time-covert channel is that receivers are continuously monitoring if 6 out of 24 frames are received within the expected time and, if not, they are discarded and an intrusion is reported. For the experimental data with 36% car traffic, the k and n values are increased to $k = 8$ and $n = 48$ in order to achieve a 99.97% success rate for legitimate transmitters and a lower value of 3.13% success rate for adversarial nodes. These rates may be considered acceptable taking into account that either $1/2$ or $2/3$ of the entire bus load is actually un-optimized, noisy traffic transmitted on top of the time-covert channel.

4.4.6 Channel data rate

Using the Arimoto-Blahut algorithm [113], [114] the maximum capacity for the time-covert channel with un-optimized (noisy) and optimized traffic is analyzed, considering a security level of 8 bits. The first step is the extraction of the channel matrix, using a free MATLAB implementation of the Arimoto-Blahut algorithm [1]. Then, the theoret-

[1] https://www.mathworks.com/matlabcentral/fileexchange/32757-channel-capacity-using-arimoto-blahut-algorithm

ical channel capacity is calculated resulting in ≈ 4.9 bits per frame. This means that the required AUTOSAR security level of 24 bits [72] can be achieved using six consecutive frames. The channel capacity for the optimized traffic can be estimated based on the tolerance value that is set. By setting the tolerance to $\rho = 10\mu s$, the covert channel is not affected by delays and the theoretical channel capacity is reduced to ≈ 4.6 bits per frame. Since, for the experiments, a security level of 7 bits is used to avoid timing collisions between frames, the time covert channel data rate is reduced to $\log_2(127/10)$, which is of 3.66 bits per frame. In what follows, the covert channel data rate is measured in bits/second (bps). Considering the effective throughput from the experiments of $1,379$ frames/second, the maximum achievable data rate using a 8-bit security level is of $6,757bps$. The variant used in the experiments with a security level of 7 bits has an effective covert channel data rate of $1,379 \times 3.66 = 5,047bps$. Considering that IDs used in the experiments have a cycle time that varies from $10ms$ to $100ms$, the effective data rate of the implemented time covert channel would be of $36 - 366bps$ for the frames. The maximum possible data rate of the channel is of $49 - 490bps$, for the maximum security level computed for the time-covert channel. By using more frames, both the busload and the time-covert channel capacity will increase. For a 54% busload, by maintaining an inter-frame space of $500\mu s$, the channel data rate is increased to $9,800bps$, for a tolerance of $\rho = 10\mu s$. Using the experimental results from both combined optimized and un-optimized traffic, as shown in Table 4.3, the tolerance allows only 65.81% and 38.70% frames to be considered legitimate. These are the results for un-optimized traffic of 18% and 36% from the total busload. The time-covert channel capacity is of $3,321bps$ for the first case and $1,953bps$ for the second case.

4.4.7 Security level

Two different security levels, one of 15-bit (i) and one of 24-bit (ii) are compared in what follows and presented in Figure 4.24. The 15-bit security level is selected since the CRC bit field inside a CAN frame has 15 bits. Even though it is used only for integrity checks and not for security purposes, its size is a good candidate for this comparison. On the other hand, the 24 bit security level is required by the AUTOSAR standards [72]. In Figure 4.24, the adversarial success rate over multiple frames is shown with blue and the adversarial success rate over k-out-of-n frames, with $n = 24$, with orange. For the 15-bit security level, the required level of protection against adversarial intervention can be achieved with 6-7 frames. According to the covert channel data rate, security level of 12 bits is reached after 3-4 frames, while the security level of 24 bits is reached after 6-7 frames. The time required for using the authentication delay is of up to $1.3ms$ for 3 consecutive frames and up to $6ms$ for 6 consecutive frames, as shown in Figure 4.25. The timings are somehow expected since the inter-frame space is set to $500\mu s$. The time for achieving the security level of 24 bits required by the AUTOSAR standards [72] is reasonable since only the MAC computation for delays is required, without any

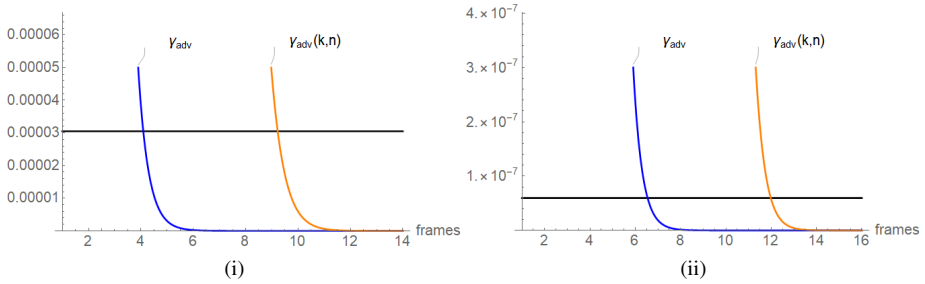


Figure 4.24: Adversary success rate for multiple frames, i.e., $\gamma_{adv}(k)$, and k-out-of-n frames, i.e., $\gamma_{adv}(k,n)$, for: (i) $k \in [1,14]$ vs. a 2^{-15} security level and (ii) $k \in [1,16]$ vs. a 2^{-24} security level ($\rho = 5\mu s$)

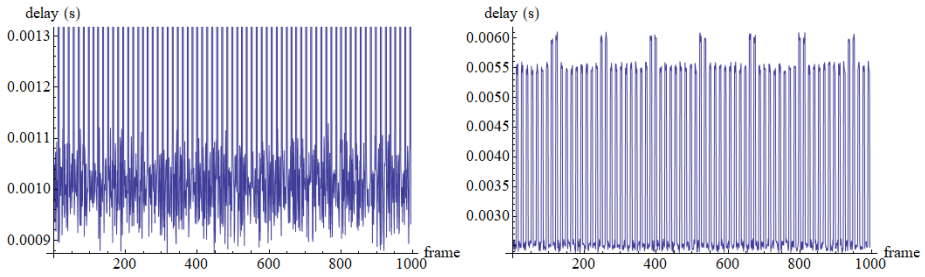


Figure 4.25: Delay between 3 (left) or 6 (right) consecutive frames

Table 4.5: Comparative performance results for covert timing channels on the CAN bus

Protocol	Throughput (single ID)	Throughput (all IDs)	BER	Security Level
TACAN-IAT [96]	22.5 bps	N/A	<1%	1 bit/frame
INCANTA [62]	57 bps	N/A	1.75%	<5 bits/frame
CANTO [30]	36-366 bps	5047 bps	0.95%	3-5 bits/frame

influence on the communication channel busload. For the combination of optimized and un-optimized channels, achieving the targeted security level of 24 bits would require more frames so using $k = 12$ out of $n = 24$ frames would be necessary. This means that it takes around $24ms$ for ensuring the security level in this case with an adversarial frame success rate of 5.7×10^{-6} and with a legitimate success rate of 96.50%, as also shown in Table 4.4.

4.5 Comparison to related works

The studies from INCANTA [62] and TACAN [96] present a covert channel by modifying the transmission/arrival times for a single carrier frame. The carrier frame from

INCANTA [62] is sent with a 100ms cycle time and transmits five bits of covert information. The carrier frame from TACAN [96] is sent with a 10ms cycle time and transmits one bit of covert information. There is an additional bit used from the frame data field in the proposal from TACAN [96] but, since it is not part of the covert channel, it is not included as part of the comparison that is done concerning the inter-arrival time. As part of the analysis, the bit-error rate (BER) is included, that is of 1.75% for INCANTA [62], but it is computed for a high-priority frame identifier. In case of the proposal from TACAN [96], the BER is less than 1%, if computed over consecutive frames. Nevertheless, depending on the busload, it can be as high as 40% for single frames. As covert channel data throughput, the authors from TACAN [96] suggest there are $22.5bps$ for their proposal. The data throughput from INCANTA [62] is of $57bps$ even though the frame cycle time is 10 times higher than the one in TACAN [96]. The increased value is due to the higher drift for the delay applied to the covert channel that is of 220ns with a tolerance of $20ns$ which means there are 5 bits covertly transmitted by each frame. Since there are only single frames used for the covert channels proposed by INCANTA [62] and TACAN [96], it cannot be predicted if, by using multiple frames, the performances would also increase. For the time covert channel from CANTO [30], that takes advantage of the optimal traffic allocation, all frames that are part of the proposal transmit covert bits. This results in a data throughput of $\sim 5,047bps$ and a BER of 0.95%. The comparison between CANTO [30], INCANTA [62] and TACAN [96] is summarized in Table 4.5.

4.6 Concluding remarks

In this chapter, several frame scheduling optimization methods were proposed and evaluated on automotive embedded platforms. These methods use the total number of frames that are required to be scheduled and their cycle times in order to maximize the inter-frame space on the CAN bus. Three of these methods, the Binary Symmetric Allocation, the Randomized Search Allocation and Greedy Allocation were shown to give very close results in the experiments to the expectations from the theoretical estimations. The differences are caused by the minimum inter-frame time which, in rare occasions, caused collisions on the bus. The Multi-Layer Greedy and GCD Allocation methods have shown experimental results that are identical to the theoretical expectations, since the minimum inter-frame space has increased. A time-covert authentication protocol was evaluated in the context of using one of the frame scheduling optimization algorithms in two scenarios, using a single sender and multiple senders. Clock synchronization and clock de-skewing were required in order to maintain the same global time between multiple transmitters. Additional results were presented in a case when the optimized and unoptimized traffic is combined. The channel data rate and security level were discussed and compared with those from related works, showing an increased performance.

Chapter 5

Clock and Voltage Fingerprinting on the CAN Bus

This chapter is based on a previous research paper of the author [31], which studies the utilization of clock skews and voltage features as fingerprint characteristics for Electronic Control Units (ECUs) in real-world vehicles. The clock skews and voltage features are used to classify and cluster individual frames belonging to ECUs from the same vehicle or from different ones. The values that were obtained through statistical analysis are presented for each vehicle, while several findings related to clock skews and voltage features are discussed. A study of the impact on physical characteristics for 1 hour driving is also presented.

5.1 Fingerprinting ECUs on CAN buses inside cars

ECU fingerprinting was performed using voltage and clock characteristics in a previous work of the author, ECUPrint [31], with the identification of 51 ECUs from 9 vehicles. Since the CAN frames transmitted in passenger cars cannot be directly linked to senders by eavesdropping the frames, the data collected for each frame was used to define a sender matrix. The information regarding frame transmission on in-vehicle networks is known by the vehicle manufacturer and its suppliers, but they are not publicly shared. Efforts in this regard have been made and some frames from in-vehicle networks have been reverse engineered and made public [1] with information regarding cycle time, data and the sender. Previous research works have extracted ECU fingerprints from cars based on clock skews [115], [116] or voltage characteristics [65] but only from one or two cars or by using an experimental setup. With our contribution, an extended number of vehicles are used for extracting ECU physical characteristics, while comparing and correlating a minimum of features required to perform the fingerprinting. Clock skews for transmitters

[1] `https://github.com/commaai/opendbc`

can be determined and used to fingerprint nodes by using the measured received time for periodic frames. It is not recommended to use clock skews alone for node fingerprinting since some impediments regarding the usage of clock skews also exist. The clock skews can be affected by unexpected delays or they may come from a different vehicle network than the one which is monitored, e.g., frames re-transmitted through a gateway ECU. For voltage-based fingerprints, using only one feature is not sufficient. Using four features, i.e., the mean voltage, max voltage, bit time and plateau time, may be enough, as we showed in [31]. Another important factor taken into account is the environmental aspect that affects both clock skews and voltage characteristics. This is also analyzed in the final part of this chapter.

Potential use cases for fingerprinting ECUs. The motivation for ECUPrint [31] is the same as for the authors of Canvas [117] who have proposed a method to map frames to specific senders using clock characteristics. The improvement from ECUPrint [31] is the utilization of both clock skews and several voltage features extracted from 9 passenger vehicles to identify 51 ECUs based on 400 different frame identifiers. A primary use-case for physical fingerprinting methods is their use for the intrusion detection system (IDS) on in-vehicle CAN networks. Nevertheless, an IDS mechanism that uses the physical characteristics was not implemented in ECUPrint [31]. The main goal of the work was voltage and clock skew collection from vehicles and their utilization as forensics to fingerprint and identify ECUs. A report published by the FBI regarding crimes [2] provides the statistics that show an increase of car thefts. Another report published by the National Crime Prevention Council [3] shows a new problem that happens after cars are stolen and that is the VIN cloning procedure. This method can be used by adversaries to forge VINs in stolen cars with other VINs from genuine cars. One way to counteract the VIN cloning issue is to store physical characteristics from vehicles for the ECUs that transmit the VIN on the bus in authorized databases and to use them for verifying their authenticity every 1-2 years during the annual technical safety inspection. This method of verifying the authenticity of ECU by physical fingerprints can also be used during traffic inspections that are required for lorries. Of course, periodically updating the physical characteristics in the authorized databases is also recommended, since the physical characteristics may change due to aging of ECU components and vehicle wiring. As a study that helps in this regard, in ECUPrint [31], the differences between physical characteristics from ECUs in the same vehicle are analyzed as intra-distances and from ECUs that are in different vehicles are analyzed as inter-distances. As already mentioned, the purpose of the analysis from this work is the utilization of physical characteristics as forensics for authenticating CAN transmitters, as a complementary measure for intrusion detection systems. All samples collected from the passenger vehicles are released as public dataset accessible from

[2] https://www.fbi.gov/news/pressrel/press-releases/fbi-releases-2020-crime-statistics

[3] http://archive.ncpc.org/resources/files/pdf/celebrate-safe-communities/NCPC-autotheft-101.pdf

GitHub and on the institution's website [4]. The voltage samples from the dataset were also used by the recent research in [118] as input for Convolutional Neural Networks (CNNs) for voltage-based fingerprinting.

Sources for fingerprinting. Automotive ECUs have various functionalities inside the vehicle, but, for CAN communication, they require a specific interface so they are connected to the CAN bus. In Figure 5.1, the external and internal physical interfaces for an ECU are shown, for the in-vehicle CAN bus context. The external interfaces are CAN-H and CAN-L wires which connect the ECU to the CAN bus through the CAN transceiver and battery voltage, VBAT, and ground, GND, lines which supply the unit. The internal interfaces required for CAN communication are the CAN serial lines, CAN-TX and CAN-RX, between the CAN controller of a microcontroller and the transceiver. Both the microcontroller and the CAN transceiver require the power supply lines, VCC-M and VCC-T, and the ground so they can operate. During operation, the CAN transceiver will convert the information received on the CAN-TX line from the microcontroller as a differential voltage on the CAN-H and CAN-L lines. The microcontroller also requires an external oscillator, OSC, that provides a clock input, CLK, for generating its internal clock for its processor and peripherals. The internal clock of the microcontroller is used to measure and control the bit time for CAN communication and, in addition, to allow the transmission of periodic frames on the CAN bus. Both the oscillator and the CAN transceiver are highlighted as genuine sources for fingerprinting ECUs using clock skews or voltage characteristics.

5.2 Related works

Considering that several research works propose identification or intrusion detection mechanisms on the CAN bus which use physical characteristics referred to as fingerprints, there are two main areas that are described in what follows. The first physical characteristics which were studied are the clock skews which can be extracted from periodic messages during active communication. The clock skews have been used as fingerprints for computers [119] and smartphones [120]. Clock skews have also been proposed as intrusion detection characteristics for nodes in wireless sensor networks [121] or access points in wireless networks [122]. With regards to automotive networks, clock skews have been proposed as physical characteristics for ECU identification for the first time in [66]. It was later shown that clock skews could be falsified by using fine-grained microcontroller timers, so using them as fingerprints is a vulnerability to cloaking attacks [67], [123]. Nevertheless, their utilization remains effective for genuine ECU identification, which will not have its clock changed over time, so the fingerprinting source does not change [117]. The environmental variation effects on clock skews, such as temperature changes, were studied in [115]. In this work the clock skews are used for identification

[4] https://www.aut.upt.ro/~bgroza/projects/ecuprint/

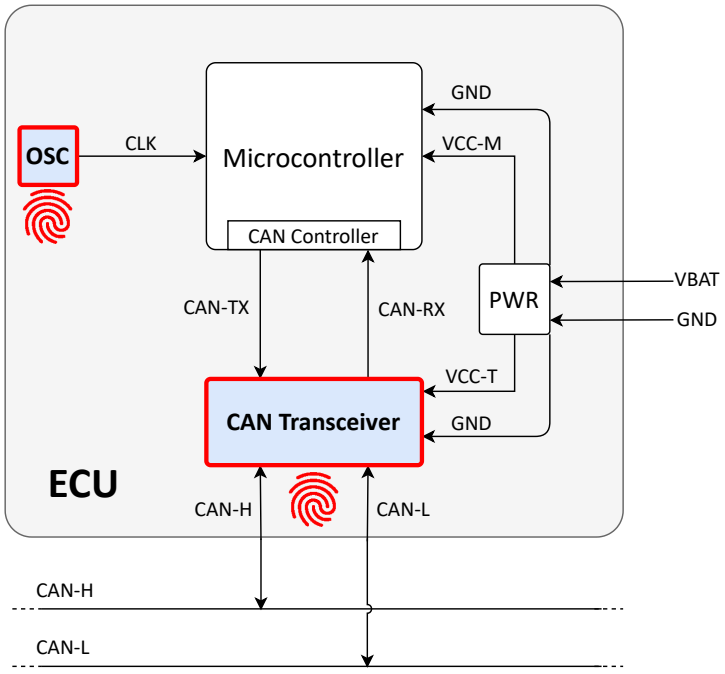


Figure 5.1: Internal block diagram of an automotive ECU with architectural components required for CAN communication

of legitimate transmitters and for intrusion detection on the Controller Area Networks.

The second research method is based on voltage characteristics that are collected as ECU fingerprints for transmitter identification or intrusion detection. Voltage features have been proposed as fingerprints for devices used for Ethernet [124] and wireless communication [125]. The first work which uses voltage features from devices that communicate on the Controller Area Network is [26]. A research paper [27] proposed the usage of specific voltage characteristics from the CAN frame including the acknowledge bit threshold for identifying the transmitter and detecting intrusions. The evaluation was performed on an Arduino-based experimental setup and on two vehicles, a Honda Accord and a Chevrolet Trax. A different approach suggested by authors in [65] is to utilize parts of dominant bits, rising and falling edges as electrical characteristics for transmitter identification. Using several statistical features for the electrical samples, both in the time and frequency domains, they trained Linear SVM (Support Vector Machine) and BDT (Bagged Decision Trees) classifiers in order to detect masquerade attacks and intrusions. They evaluated their proposal using an experimental setup with 12 nodes and two cars, a Kia Soul and a Hyundai Sonata. Authors from [126] propose the usage of single CAN frames for identifying transmitters and reporting intrusions. The voltage fea-

tures used for training a classification matrix are extracted from the CAN frames and are based on samples collected from dominant or recessive bits after rising edges or falling edges. Using the Mahalanobis distance, their algorithm verifies if the voltage sample collected in real time on the vehicle bus matches the intended origin in the classification matrix. The algorithm was both trained and evaluated using data from an experimental setup with 10 nodes and two networks from real-world vehicles, a CAN bus with 5 ECUs from Nissan Sentra and a CAN bus with 4 ECUs from a Subaru Outback. Separation of electrical characteristics in different groups before training and testing is also suggested in [127]. These groups contain samples for rising edges, falling edges and dominant bits, which are analyzed using statistical features such as the mean value, standard deviation or variance. In this work, the voltage samples are collected using a PicoScope 5204, while the algorithm for detecting intrusions is implemented in Python. Evaluation is done on an Arduino-based experimental setup with 6 nodes and two vehicles, a Fiat 500 and a Porsche Panamera, both with 6 ECUs connected on the vehicle bus. This work was extended in [128] with focus on temperature variation influence on voltage data. Transmitter identification using only the rising and falling edges is proposed in [129] where the voltage samples are collected with a PicoScope 5000 series having the resolution set to 8 bits. A broad analysis regarding the voltage sampling method selected for collecting the data is done by authors in [130] for their previously proposed intrusion detection algorithms [127, 128, 129]. The effects of different sampling methods are emphasized by verifying the signal quality and verifying which option works better by evaluating it on a real vehicle. They concluded that the average of the voltage samples is the better choise for intrusion detection systems. A light version of an intrusion detection method that uses voltage characteristics and is intended for low resource automotive microcontrollers is proposed by the same authors in [131]. The voltage sample dataset collected from a CAN bus prototype was used as input for generation of the average and standard deviation statistical features in [132]. The CAN bus prototype includes 9 different nodes that communicate while one of them is dedicated for source identification. The authors of [132] suggest that their proposal has an improved performance with regards to false positives and false negatives compared to the proposal from [27].

Since all previous proposals use the voltage samples to fingerprint each transmitter, the authors from [133] propose a network layout fingerprinting method based on time domain reflectometry. This method allows a supervising node to determine if nodes were added to the network or removed from the network. Bit time monitoring [134] is a new proposal that makes use of both voltage and time characteristics for transmitter identification with the use of classifiers trained with statistical features of the bit time such as the mean value and its standard deviation. Since most of the research papers propose the utilization of voltage characteristics for source identification or intrusion detection, they can be used for detecting attackers during spoofing attacks or bus-off attacks as well, as shown in [135]. The addition of an independent node that does not communicate on the CAN bus but performs active monitoring for intrusions based on voltage characteristics

is proposed in [136]. The performance of the intrusion detection capability is suggested as being more than 97% as true positive rate. Using an experimental setup with 19 nodes where 18 are genuine and 1 is adversarial, the authors from [137] propose a reinforcement learning authentication method using CAN-H and CAN-L voltage samples for all frames and arrival times for cyclic frames. There are 14 cyclic messages considered while 4 are on-event with a busload of $\sim$38%. The performance improvements for this authentication method, if compared with existing proposals, are both in regards to false positives, of 0.8%, for periodic frames and false negatives, of 4.4%, for on-event frames. Even though the voltage information is feature-rich, one research work has proposed and evaluated a method of evading voltage-based intrusion detection methods [138]. Their proposal is a masquerading attack that can manipulate the voltage fingerprints for legitimate nodes using two adversarial nodes, an attacker and an accomplice. The authors also propose a method that needs to be used for re-training the voltage-based intrusion detection system in order to validate that all transmitters are genuine. Mitigation of voltage fingerprinting intrusion detection methods from [139] are evaluated against existing defense proposals.

5.3 Physical layer data collection from real vehicles

Considering the background for collecting fingerprint data from ECUs based on fingerprint sources like oscillators and CAN transceivers, the focus for data collection activities is to sample data required for determination of clock skews and voltage features. Clock skews can be calculated for periodic CAN messages, while voltage features can be determined from the CAN physical bus using both CAN-H and CAN-L communication lines. Although previous research papers have used either clock skews or physical characteristics to fingerprint ECUs from vehicles, there is no previous paper which shows both physical characteristics for the same ECUs and in the same vehicles. The cars that were used for data collection, the number of identified ECUs and frame identifiers, the average busload and environmental conditions for the data collection are summarized in Table 5.1. The summary of frames collected in [31] for determination of the clock skew for each ECU and CAN bits collected for determination of voltage features for each ECU is shown in the same table. There are 9 passenger vehicles from which more than 220,000 bits were collected as voltage data and more than 8 million frames are collected for clock skew determination. Accordingly, the comprehensive dataset can be used for analysis of forensics or as basis for intrusion detection systems that can spot physical characteristics differences between distinct automotive control units. Since the CAN matrix for each vehicle which contains information regarding frame transmitters, the frame periodicity and signals contained in each frame are unknown to the public, identification of ECUs from passenger cars is considered to be a challenging activity. By determining the clock skew for each frame identifier, the frames can be grouped using the same, or very similar clock skews, as part of the same ECUs. The same goes for the voltage data which, even

Table 5.1: Summary of identified ECUs based on evaluated data in [31]

Vehicle	No. ECUs	No. IDs	Busload	Temp.	VBAT	Collected frames(skew)	Collected bits(voltage)
Honda Civic	6	43	31%	9 °C	12.8 V	1,039,512	40,073
Opel Corsa	4	29	23%	8 °C	13.9 V	442,992	9,187
Hyundai i20	7	40	35%	12 °C	14.2 V	616,296	17,767
Dacia Duster	3	12	14%	10 °C	14.4 V	247,154	9,086
Dacia Logan	6	46	14%	10 °C	12.6 V	629,662	31,579
Hyundai ix35	6	26	45%	9 °C	13.5 V	847,161	23,104
Ford Fiesta	6	46	51%	5-7 °C	14.9 V	2,243,359	43,861
Ford Kuga	9	70	65%	9 °C	13.7 V	1,233,545	28,024
Ford Ecosport	4	87	43%	9 °C	15.0 V	759,421	22,808
Total	**51**	**399**	-	-	-	**8,059,102**	**225,489**

though it is affected by the environmental factors, can be used to separate between the sender ECUs. For voltage data, the changes over time are analyzed using data captured in two cars. The data was collected from these vehicles at startup and, later, after they were driven for one hour.

Using mixed physical characteristics as fingerprints. Both timing and voltage characteristics are necessary for ECU fingerprinting, as explained in what follows. Using only clock skews for determining which ECU is the transmitter is easier than using voltage characteristics even though it requires multiple frames for a stable value. Authors from a previous research paper [67] show that clock skews can be easily adjusted and legitimate ECUs can be replaced by adversarial nodes capable of faking the clock skews. So, using clock skews alone, is proven to be insufficient. Using voltage features is problematic because it is somewhat hard to collect the voltage data considering the CAN bit rate and the capabilities for ADC sample rates. This means that, even if voltage features cannot be falsified as easily as clock skews, they require equipment with improved capabilities for collecting voltage samples. Another challenge for voltage-based fingerprinting is related to the selection of voltage features. By using four features that are mean and maximum voltages together with the bit and plateau times, ECUs can be classified based on frames, as shown in the analysis part that follows.

Pros and cons for fingerprint sources. Now the pros and cons for using clock skews or voltage features for ECU fingerprinting are discussed. A summary of the advantages and disadvantages of using clock skews or voltage features is shown in Table 5.2. As mentioned earlier, it is easy to collect frames for determination of the clock skews that may already be preserved through gateways from the vehicle. Unfortunately, it is also easy to falsify the clock skews which cannot be determined for on-event frames. In case frames are affected by arbitration and arrive with delays on the vehicle bus, it will make clock skew estimation harder. Another disadvantage of clock skew determination is that a small number of frames is not sufficient. This means that for an accurate clock skew determination, a larger number of frames is required. In contrast with the clock skews, the voltage data is harder to collect since it requires both access to each physical bus and equipment that uses high sample rate ADCs to allow the collection of voltage feature data for single bits. It is harder to falsify voltage data because there are multiple features that can be extracted from a few hundred or thousand samples from a single bit. This means

Table 5.2: Pros and cons for skew-based and voltage-based fingerprinting on CAN

Fingerprint method	Advantages	Disadvantages
Clock Skews	✓easy to collect ✓may be preserved through gateways (possible to retrieve from distinct buses)	✗easier to forge ✗do not work for on-event frames ✗affected by arbitration and processing delay ✗require many frames for estimation
Voltage Features	✓harder to forge ✓single bit/frame is sufficient ✓feature rich fingerprint	✗harder to collect, may require high sampling rate ADCs ✗require physical access to the same bus

that using both clock skews and voltage data is required for fingerprinting ECUs and this is shown as part of the analysis done on both physical characteristics using information from the dataset.

The frame timings and voltage data from [31] are collected from 9 different passenger vehicles with various body types. The Hyundai i20, Ford Fiesta, Opel Corsa and Dacia Logan have a hatchback body-style. The Honda Civic is a sedan and Dacia Duster, Hyundai ix35, Ford Kuga and Ford Ecosport have a Sport Utility Vehicle (SUV) configuration. As already mentioned, all vehicle models from which clock skew and voltage samples were collected are shown in Table 5.1. One more thing worth mentioning is that vehicles from which the data is collected for ECU fingerprinting have a worldwide diversity because Ford is a United States manufacturer, Hyundai and Honda designed their vehicles in Asia, while Opel and Dacia are European vehicle manufacturers. The internal bus networks that were accessible were those connected to the OBD-II interface, as shown in Figure 5.2 (i)–(ix). The number of ECUs that are referenced in the Figures is based on the classification and clustering using voltage features and clock skews. Based on the analysis, the ECU separation cannot be done using only the clock skews. This is because some clock skew fingerprints are determined for ECUs connected to a different vehicle network that are forwarded through gateways connected to the vehicle bus used for data collection. By searching examples of vehicle networks that are shared on public forums and web pages, the ECUs connected to the network with OBD-II interface from some vehicles were identified. The ECU naming for the cars where the information was not found online is represented using a generic name, $\mathrm{ECU}_i, i = 1..n$, as shown in Figures 5.2 (i), (ii), (iii), (v), (vi), (ix). For Honda Civic there are 6 different ECUs identified connected to the OBD-II CAN bus, while Opel Corsa and Ford Ecosport has only 4 ECUs. The Hyundai i20 has 7 ECUs connected to the vehicle bus used for diagnostics, while Dacia Logan and Hyundai ix35 have only 6 ECUs. The wiring diagrams for Dacia Duster were found on a public forum [140]. Using this information, the nodes connected to the OBD-II accessible network are identified. Based on the information found online, the ECUs from the Dacia Duster are the ABS control unit (ABS), the injection system control unit (INJ) and the front/rear torque distribution control unit (FRTD). They are shown in Figure 5.2 (iv). Based on the wiring diagrams from [141], the 6 ECUs that communicate on the OBD-II CAN bus from the Ford Fiesta were identified. The Ford Fiesta nodes shown in Figure 5.2 (vii) are the gateway module (GWM), sync module (APIM), headlamp control module (HCM), powertrain control module (PCM), body control mod-

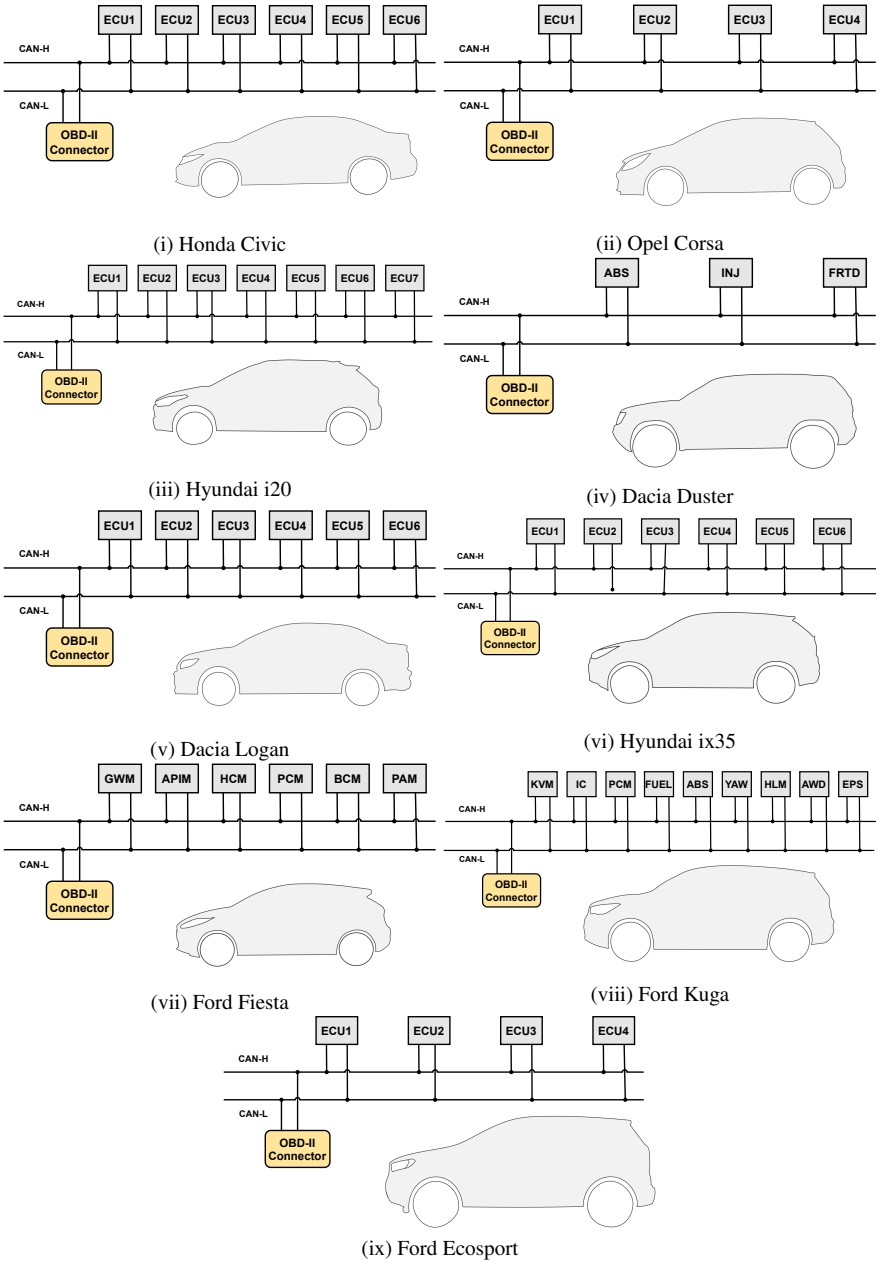


Figure 5.2: Illustration of CAN networks with OBD-II access for Honda Civic (i), Opel Corsa (ii), Hyundai i20 (iii), Dacia Duster (iv), Dacia Logan (v), Hyundai ix35 (vi), Ford Fiesta (vii), Ford Kuga (viii) and Ford Ecosport (ix)

ule (BCM) and parking aid module (PAM). Information related to the Ford Kuga wiring networks is available on a webpage [142] that was used to specify the ECUs from the Ford Kuga vehicle. The Ford Kuga nodes from Figure 5.2 (viii) are the keyless vehicle module (KVM), instrument cluster module (IC), powertrain control module (PCM), fuel additive system module (FUEL), ABS module (ABS), yawrate sensor (YAW), headlamp leveling module (HLM), all-wheel drive control unit (AWD) and electrohydraulic power steering module (EPS).

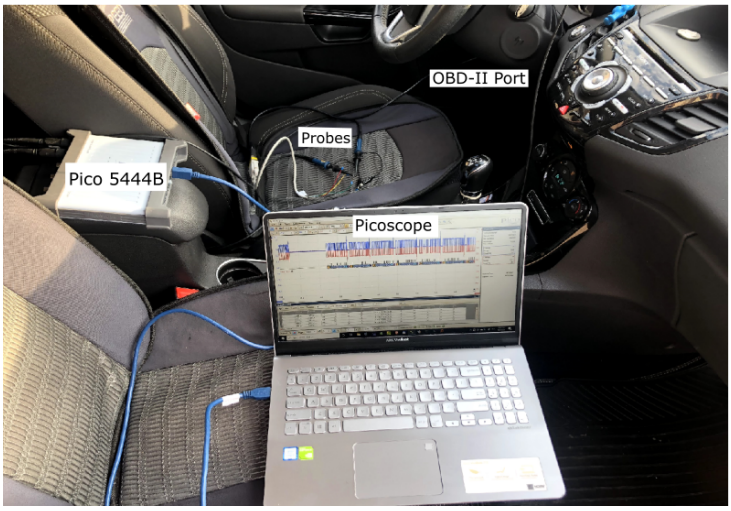
5.3.1 Data collection setup

Now, the tools used for collecting the frame and voltage data from the vehicles are described. In order to collect the traffic data from the vehicle networks with OBD-II, a CANcaseXL produced by Vector was used. Vector is one of the biggest companies to provide hardware equipment for the automotive industry in order to interface laptops and computers to vehicle buses such as CAN, CAN-FD or FlexRay. The software tools from Vector that are already available can be used either to collect or inject data on existing networks. One example is the XL Driver Library, which is an open source library from Vector which allows communication on the CAN bus from a PC or laptop using Vector hardware devices. A custom application that uses the XL Driver Library was built, since the intention was to collect the vehicle bus data together with the internal timestamps of the CANcaseXL device. In order to interface the CANcaseXL to the CAN bus from the vehicles, the baudrate was configured to match the one from the vehicles of 500Kbps (same for all passenger vehicles). In Figure 5.3 (ii), the OBD-II diagnostic port from the Ford Fiesta vehicle is shown. This interface was used to collect CAN frames and voltage samples. In the same figure, the CAN lines and the ground (GND) line are emphasized since they were the physical access point required to perform data collection. In order to interface the CANcase XL to the OBD-II port, a cable adapter was used with both the DB9 and OBD-II interfaces which is the required pinout for the vehicle connector and the Vector device. The frame data was collected in 5-10 minutes sessions, after the vehicle was started. Frames that are sent at vehicle startup were not collected since there may be initialization frames or on-event frames for which the clock skew cannot be determined.

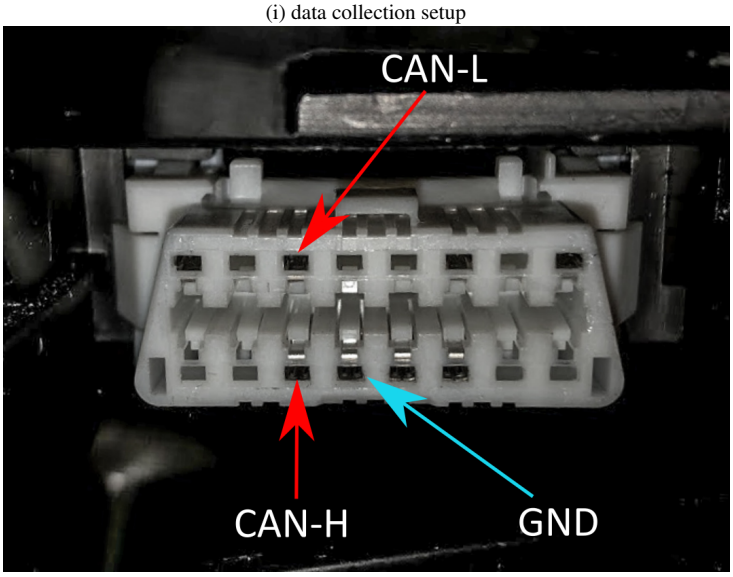
In order to collect the voltage data from the vehicle networks, a 5000 Series Picoscope produced by Pico Technology was used. The probes were attached to the required electrical interfaces, i.e., CAN-H, CAN-L and GND, that were accessible through an additional cable. The additional cable has a DB9 interface on one side and on the other it separates individual CAN wires and the GND line. Using the PicoScope 6.14 software tool on a laptop, collected multiple files that contain voltage samples of the CAN network were collected from each vehicle. The data collection setup installed in the Ford Fiesta vehicle is shown in Figure 5.3 (i). The CAN serial decoding option from the tool was used with the baud rate set to 500Kbps, so the CAN messages saved in each file were visible in the tool with the entire data interpretation. Since the frame identifiers were

recorded in the traffic data collection step for clock skew, the frames expected for voltage data collection were already known. As settings for the PicoScope software tool, the voltage range was configured to ± 5V since a higher value than 4.5V for the CAN-H line is not expected. The sample rate was configured to 500 MS/s because 2ns are required for each sample while using two channels for collecting voltage data on CAN-H and CAN-L lines. A maximum frame duration of around $270\mu s$ is expected since the bit rate for passenger vehicles is of 500Kbps as recommended by SAE J2284-3 [89]. Considering the bit rate and frame duration for the passenger vehicles, the windows for capturing voltage data were configured to 4ms, so, voltage samples for a dozen of frames can be collected in each capture window. For each channel, the hardware resolution was set by default by the tool to 8 bits but, using the resolution enhancement option, it was extended for a more accurate sampling to 12 bits. Even though the cost for a 5000 Series Picoscope is somewhat high, the Xilinx XC6SLX25 FPGA or Texas Instruments ADC08D502 high performance ADCs, which are similar to its key components, do not have an acquisition cost of more than 100$ (US Dollars). This means that the design of a tool that meets the capability needs for ECU fingerprinting is not as expensive as a 5000 Series Picoscope.

Each file which was collected with the Picoscope tool has 267 independent windows with both voltage samples and CAN frame interpretation using the serial decoding option. The frames for which the voltage data was collected were checked offline since the CAN-H threshold and the hysteresis were automatically set by the tool, allowing the data interpretation to be visualized for each CAN frame. The sample windows were collected based on a trigger on the CAN-H line to capture the SOF bit for a frame which is a transition from bus idle (recessive) to dominant. The voltage threshold for the transition was set to 3V. Since the files are stored with psdata extension, a PicoScope software tool proprietary format, these files could not be used as input for analyzing the voltage data. Using the PicoScope software tool, each sample window was exported in csv (comma separated values) files that contain the timestamp and voltage data sampled every $2ns$. Since the csv files did not contain the frame identifiers, they had to be manually exported from the PicoScope tool using the CAN serial decoding part that contains the timestamp and CAN frame data from each sample window as separate data files. In order to combine the information from the exported files, a python script was developed and used to extract frame voltage data from the csv files as separate files using the timestamp information from the data files. The combined files were grouped in separate folders, each folder named with the CAN identifier for the frames for which the voltage data was collected. In order to capture voltage data for individual CAN bits, the voltage samples for data and CRC areas from the combined files were exported first. Only data and CRC areas were used because the intention was not to use voltage data from the arbitration field, since multiple nodes may communicate at the same time on the bus. From the voltage samples for data and CRC bitfields, the voltage samples for individual dominant bits which are isolated between recessive bits were extracted. In this way, the dataset with voltage samples for isolated dominant CAN bits from 9 passenger vehicles was created.



(i) data collection setup



(ii) OBD-II port

Figure 5.3: Data collection setup from Ford Fiesta (i), the OBD-II port with specified CAN differential lines and vehicle ground line (ii)

5.4 Theoretical framework

In this section, the details related to how the clock skews and voltage features were computed are detailed. Some examples of the physical characteristics are presented before the definition of the intra-distances and inter-distances.

5.4.1 Clock skews

Using the notations from [112], the information regarding clocks, clock offset, clock skew and clock drift is used in order to define the theoretical framework. The system clock is defined as a continuous function $\mathbb{C} : \mathbb{R} \to \mathbb{R}$ that is twice differentiable. The offset of a clock is the difference between the expected time and the real time, as defined in Equation 5.1. The first derivative of the clock is the clock frequency $\mathbb{C}'(t)$, while the second derivative of the clock $\mathbb{C}''(t)$ represents the clock drift. The clock skew is the difference between the first derivatives of the clocks, as shown in Equation 5.2. The clock drift is the difference between the second derivatives of the clocks, as shown in Equation 5.3.

$$\mathbb{C}^{A}_{\text{offset}} = \mathbb{C}^{A}(t) - t \tag{5.1}$$

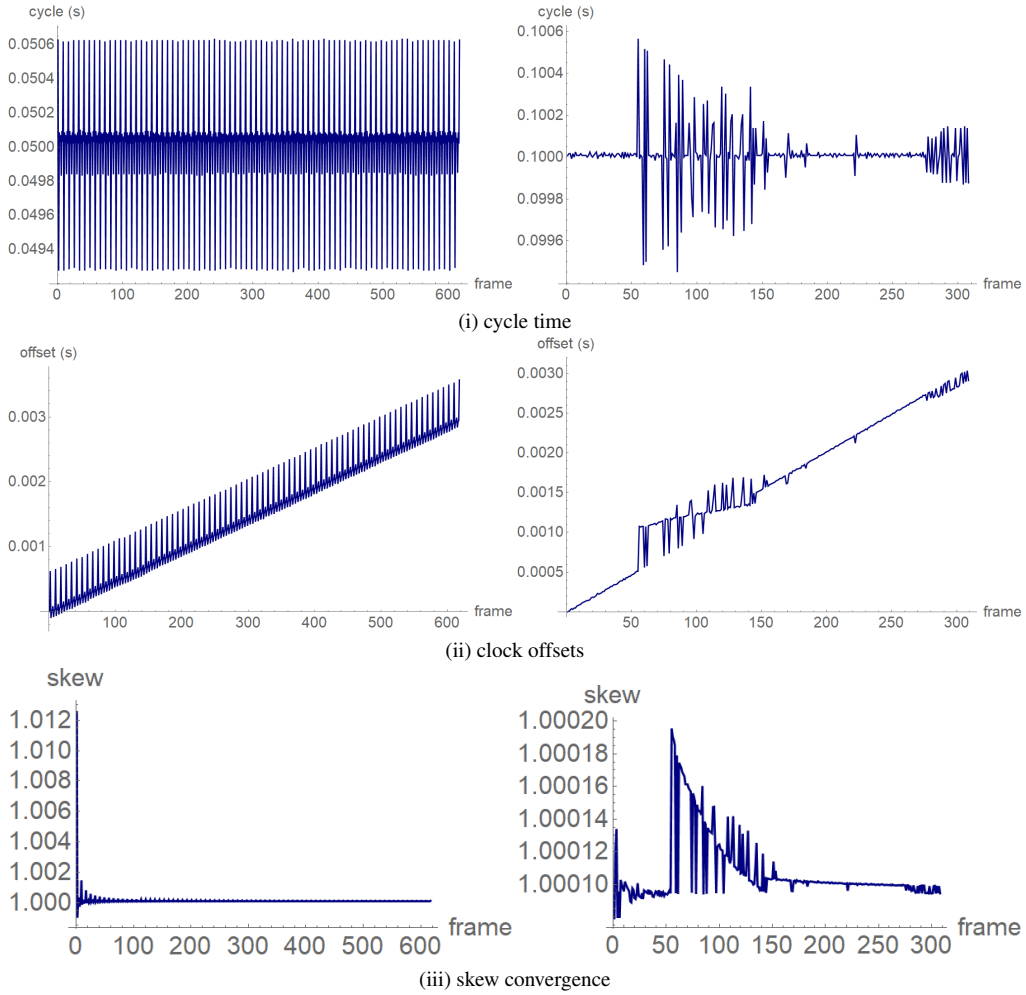
$$\mathbb{C}^{A,B}_{\text{skew}} = \mathbb{C}'^{A}(t) - \mathbb{C}'^{B}(t) \tag{5.2}$$

$$\mathbb{C}^{A,B}_{\text{drift}} = \mathbb{C}''^{A}(t) - \mathbb{C}''^{B}(t) \tag{5.3}$$

For periodic frames, the expected transmission time is defined as $t = i \times \delta_{id}$, where δ_{id} represents the cycle time for each id. The clock skew for periodic frames from the receiver standpoint can be estimated as defined in Equation 5.4 using the timestamps t_i and t_j for the received frame. Even though the recommendation from [112] is to approximate the clock skew using complex algorithms, the values computed using Equation 5.4 provide a clear separation between ECUs, if enough frames are collected to compute the mean or median values over the reception times.

$$\mathbb{C}_{\text{skew}}(id) \approx \frac{t_j - t_i}{(j - i) \times \delta_{id}} \tag{5.4}$$

The cycle time (i), clock offsets (ii) and clock skew convergence (iii) for two distinct IDs from a 30 second time window are shown in Figure 5.4. On the left side of the figure, the information is shown for an ID with a cycle time of $50ms$. On the right side of the figure, the information is shown for an ID with a cycle time of $100ms$. Since the expected reception time for the frames is based on their cycle time and the measured reception time is the one reported by the CANcaseXL device, the variation for each signal can be determined. The cycle time shows a variation of $\pm 0.6ms$ for both IDs but with a different distribution between them. The ID on the left side has more frequent variations from its theoretical cycle time, while the ID on the right has less. The clock drift has similar variations for both IDs, but somehow, the slope for the clock offset increment is similar. The clock skew for the ID on the left looks stable starting from the first 100 frames, while the clock skew for the ID on the right varies from 1.00010 to 1.00020 and back. This happens due to lost arbitration that delays the frame transmission time on the bus and affects its reception time by modifying the measured clock offset and the



(i) cycle time

(ii) clock offsets

(iii) skew convergence

Figure 5.4: Cycle time (i), clock offsets (ii) and convergence of the skew (iii) for two IDs with 50ms and 100ms cycle (left vs. right)

clock skew, as shown in Figure 5.4. This means that, even though clock skew can be easily computed based on the expected and measured reception time, there is a minimum of frames required to be received, so the clock skew value is reliable. Otherwise, lost arbitration on the bus or internal delays caused by the transmitter for specific frames would, in the end, lead to an erroneously estimated clock offset and clock skew for that ID. Thereby, using only clock skews for real-time intrusion detection is not possible because a certain number of frames is required to estimate a reliable clock skew for each frame.

5.4.2 Voltage features

The nodes are connected to the bus using CAN transceivers which provide unique features for the voltage signals that they command. The CAN-H and CAN-L voltages can be analyzed if enough samples are available and features can be extracted from them. The differences between CAN differential voltages provided by nodes are also caused by the CAN transceiver manufacturers which provide specific tolerances relative to how each transceiver controls the voltage signals. This information is usually available in the CAN transceiver datasheet. The voltage characteristics are also influenced by the receiver nodes connected on the same bus [34], but also by the cable length or the number of stubs from the main cable to the nodes. The mean voltage, maximum voltage, bit time and plateau time are used as voltage features in the analysis. These are extracted from the voltage samples for individual dominant bits which are isolated between recessive bits.

Some of these voltage features have been proposed in previous works. The mean voltage was proposed as voltage characteristics for ECU fingerprinting by authors in [127], [129] and [143]. The maximum voltage level was used as voltage feature in [65], [127], [129] and [143]. The bit time duration was extracted and used for transmitter fingerprinting by authors in [134], [144], [145] and [146]. The plateau time has not been proposed in previous works, even though it helps differentiate between transmitters. This is why it is considered as the fourth voltage feature in the analysis. Using them as individual voltage features is somewhat satisfactory for fingerprinting ECUs, but, considering overlaps, combining the information improves the ECU separation. The first paper that proposed signal characteristics for fingerprinting nodes on Controller Area Networks is [26]. There are several works which have followed with improvements on the separation between fingerprinted ECUs using classification algorithms and machine learning on specific voltage features [66], [65], [127]. Studies related to the impact of the environment on voltage characteristics for ECUs were examined in [128], [129], [126]. Most of the works use the voltage characteristics to fingerprint each sender. A different approach is proposed by authors in [133] who perform a fingerprint of the network layout in order to detect if nodes are removed or added to the network. One of the voltage features that is used is the mean voltage while the bus is in a dominant state, when the differential voltage between CAN-H and CAN-L is of $\sim 2V$. Even though mean voltage can be computed over the rising edge and falling edge of each bit, in this case it is computed only using the voltage samples from the dominant bit plateau area. Based on the voltage samples for isolated dominant bits, $s_1, s_2, ..., s_\ell$, the definition of mean voltage and maximum voltage is formalized in Equations 5.5 and 5.6. The value for ℓ is set to $2{,}000$ with one sample available at every $2ns$, since the capture window is of $4ms$. Setting the value of τ to 150 allowed us to compute the maximum voltage after reaching the bit plateau and mean voltage on the dominant bit plateau area. Considering the values of $\alpha \leq \ell/2, \beta > \ell/2$ and σ as the configured sample time of $2ns$ and the voltage samples for isolated dominant bits $s_1, s_2, ..., s_\ell$, the definition of bit time and plateau time is formalized in Equations 5.7 and 5.8. The value of ϵ, which is the threshold for measuring the bit time and plateau

time, is set to $20mV$ based on the differences observed in the experimental data.

$$\mathbb{V}_{\text{mean}}(id) = \text{mean}\left\{ s_i : i = \ell/2 - \tau..\ell/2 + \tau \right\} \tag{5.5}$$

$$\mathbb{V}_{\text{max}}(id) = \max\left\{ s_i : i = 1..\ell/2 - \tau \right\} \tag{5.6}$$

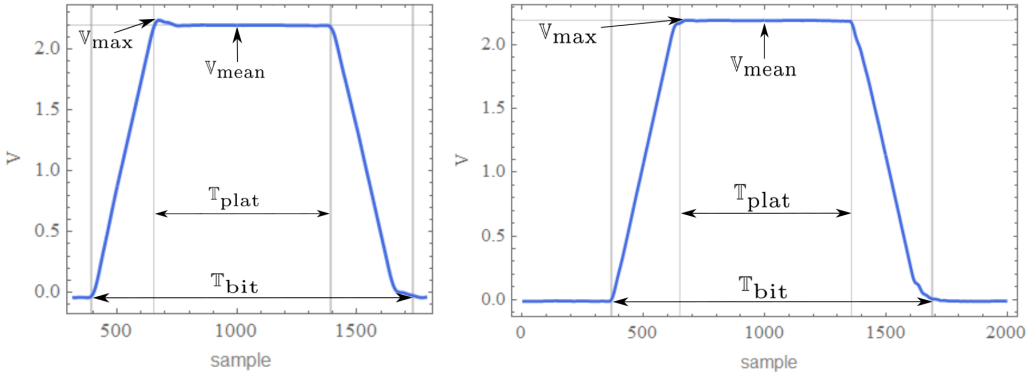
$$\mathbb{T}_{\text{bit}}(id) = \min_{\alpha,\beta} \left\{ (\beta - \alpha)\sigma : |s_\alpha| \leq \epsilon, |s_\beta| \leq \epsilon \right\} \tag{5.7}$$

$$\mathbb{T}_{\text{plat}}(id) = \max_{\alpha,\beta} \left\{ (\beta - \alpha)\sigma : |s_\alpha - \mathbb{V}_{\text{mean}}(id)| \leq \epsilon, |s_\beta - \mathbb{V}_{\text{mean}}(id)| \leq \epsilon \right\} \tag{5.8}$$

The equations are applicable only for individual dominant bits, isolated between recessive bits, as earlier mentioned. In case there are multiple dominant bits isolated between recessive bits, the equations can be adapted to this use case. By following the bit stuffing requirements from the CAN standard, after transmission of 5 consecutive bits with the same polarity, either recessive or dominant, the 6th bit must have a distinct polarity. The value for the plateau time for multiple bits can be computed as the division of the entire plateau time to the number of consecutive dominant bits. The value for the bit time can be computed as the total time for the bits from which the plateau time for all bits except one is subtracted. The values for the mean voltage and maximum voltage are computed in the same way as for individual bits. A visual representation of the voltage features is provided in Figure 5.5. The features were extracted using voltage samples from two bits that originate from different ECUs. The values determined for ID 171 (i) from Ford Ecosport are $\mathbb{V}_{\text{mean}} = 2.195mV$, $\mathbb{V}_{\text{max}} = 2.236mV$, $\mathbb{T}_{\text{bit}} = 2.686\mu s$, $\mathbb{T}_{\text{plat}} = 1.478\mu s$, while those for for ID 428 (ii) from Hyundai ix35 are $\mathbb{V}_{\text{mean}} = 2.191mV$, $\mathbb{V}_{\text{max}} = 2.195mV$, $\mathbb{T}_{\text{bit}} = 2.640\mu s$, $\mathbb{T}_{\text{plat}} = 1.417\mu s$. The information that is represented on the Y axis from Figure 5.5 is the voltage data while the samples are shown on the X axis. Since there are $\sim$1300 samples from the beginning of the rising edge to the end of the falling edge, the bit time is expected to be of $\sim 2.6\mu s$ which is confirmed by the voltage sample values for each ECU.

5.4.3 Intra-distances and inter-distances

Having the equations for clock skew determination and voltage features in place, the intra-distances and inter-distances are defined. This supports the measurements for the differences between physical characteristics for frames transmitted by the same ECU or by different ECUs. Since the timings and voltage samples are numeric values, the one dimension Euclidian distance is used, i.e., $d(u, v) = \sqrt{(u - v)^2}$, to measure the differences for the same features. This can be applied to multiple features, if required, by extending the number of dimensions. The values for intra-distances and inter-distances



(i) ID 171 (ii) ID 428

Figure 5.5: Collected voltage levels for IDs from distinct ECUs

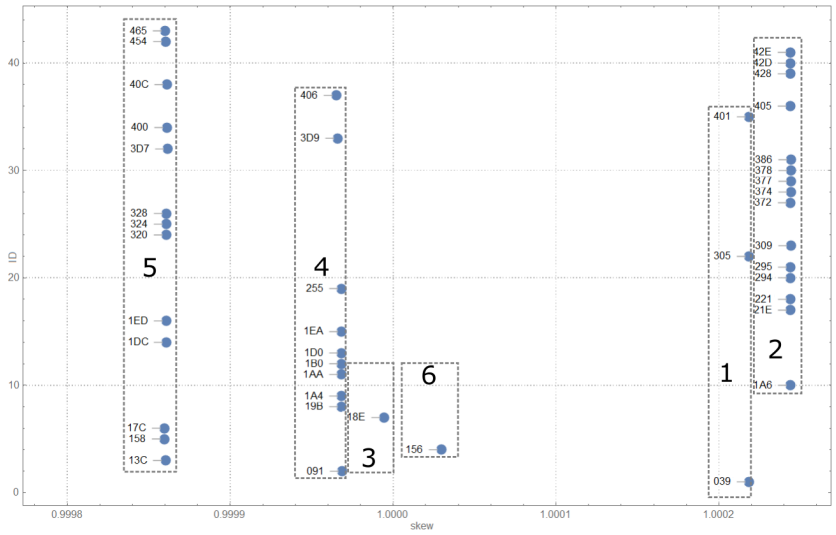


Figure 5.6: Separation for ECUs in the Honda Civic based on skews

based on the sample data can be computed as $\mathcal{D}_{\omega}^{\text{skew}}$ for clock skews, $\mathcal{D}_{\omega}^{\text{mean}}$ for mean voltage, $\mathcal{D}_{\omega}^{\text{max}}$ for max voltage, $\mathcal{D}_{\omega}^{\text{bit}}$ for the bit time and $\mathcal{D}_{\omega}^{\text{plat}}$ for the plateau time where $\omega \in \{\text{inter}, \text{intra}\}$. Using this information, the definition of the intra-distances and inter-distances is formalized in Equations 5.9 and 5.10:

$$\mathcal{D}_{\text{intra}}^{\alpha}(i) = \left\{ d(\varphi(id'), \varphi(id'')) : \forall id', id'' \in \text{ECU}_i, id' \neq id'' \right\} \tag{5.9}$$

$$\mathcal{D}^{\alpha}_{\text{inter}}(i,j) = \left\{ d(\varphi(id'), \varphi(id'')) : \forall id' \in \text{ECU}_i, \forall id'' \in \text{ECU}_j \right\} \tag{5.10}$$

In these equations, (α, φ), are the clock skew and voltage feature fingerprints, i.e., $(\alpha, \varphi) \in \{(\text{skew}, \mathbb{C}_{\text{skew}}), (\text{mean}, \mathbb{V}_{\text{mean}}), (\text{max}, \mathbb{V}_{\text{max}}), (\text{tbit}, \mathbb{T}_{\text{bit}}), (\text{tplat}, \mathbb{T}_{\text{plat}})\}$. The values of $i, j = 1..n$ cover the n ECUs determined based on the experimental results. This means that the intra-distances are determined for frames transmitted by the same ECU while inter-distances are determined for frames transmitted by different ECUs.

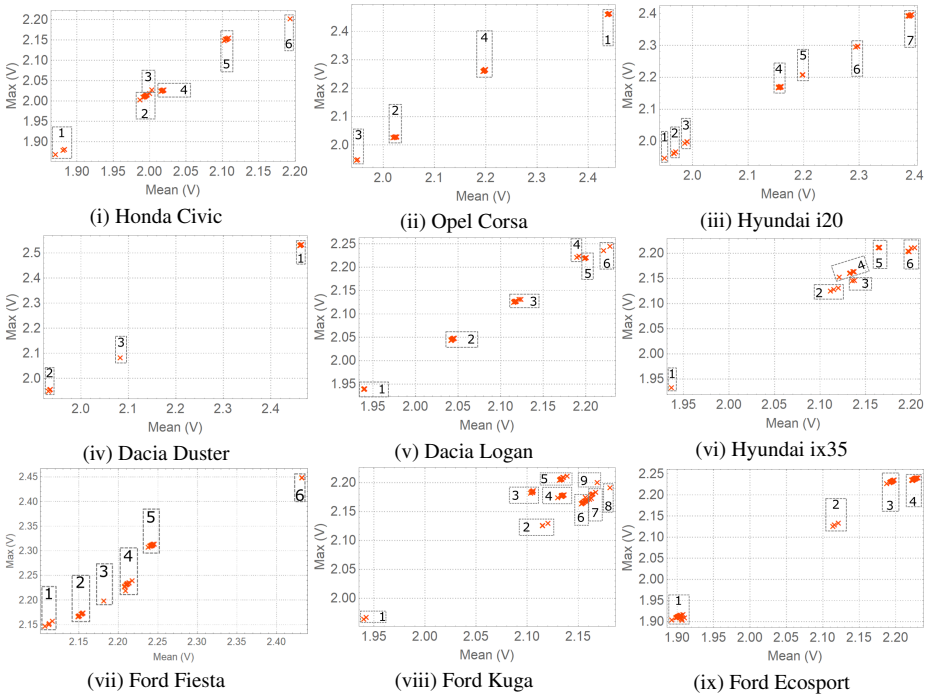
5.5 Interpretation of experimental data

This section details the resulted clock skews and voltage features for all CAN frames from the passenger vehicles. The grouping of frames in ECUs and the findings are discussed in this section. The variation of the clock skews and voltage features after two vehicles were driven for one hour is also explored at the end of this section.

5.5.1 ECU separation based on clock skews and voltage features

In what follows, the ECU separation is done for each vehicle using the computed clock skew data and voltage characteristics that were determined for each cyclic frame. Taking all results into consideration, the indication is that using single voltage features or only the clock skews is insufficient for a clear separation of the frame transmitters. A drawback relevant for clock based fingerprinting is that frames transmitted by several nodes, considered by the voltage fingerprint, would have different clock skews. This means that those transmitters are actually gateways for those frames from other in-vehicle buses. A downside for voltage based fingerprinting is that sometimes the extracted features are similar for different transmitters. These limitations are explained, whenever they are valid, for specific ECUs. The results for each physical characteristic are shown inside Tables 5.3–5.11, where the table columns represent the following information. The ID column contains the hexadecimal representation of the frame identifier, while the ECU column contains the ECU number grouping the IDs. The frame cycle time, measured in milliseconds, is written in the Cycle column, while the $\mathbb{C}_{\text{skew}}$ column contains the determined clock skew. The $\mathbb{V}_{\text{mean}}$ and $\mathbb{V}_{\text{max}}$ columns contain the mean and maximum voltage, measured in Volts, while $\mathbb{T}_{\text{bit}}$ and $\mathbb{T}_{\text{plat}}$ columns contain the bit time and plateau time, measured in microseconds.

In the **Honda Civic** passenger vehicle there are 6 ECUs found based on 43 cyclic messages transmitted on the CAN bus. By using clock skews for ECU separation, there are 6 clusters of IDs determined on the Y axis with the same values as shown in Figure 5.6. The voltage separation using mean voltage and maximum voltage features is clear for 3 ECUs, but it is challenging for the remaining three, ECU_2, ECU_3 and ECU_4. The



(i) Honda Civic (ii) Opel Corsa (iii) Hyundai i20

(iv) Dacia Duster (v) Dacia Logan (vi) Hyundai ix35

(vii) Ford Fiesta (viii) Ford Kuga (ix) Ford Ecosport

Figure 5.7: Mean-max voltage separation

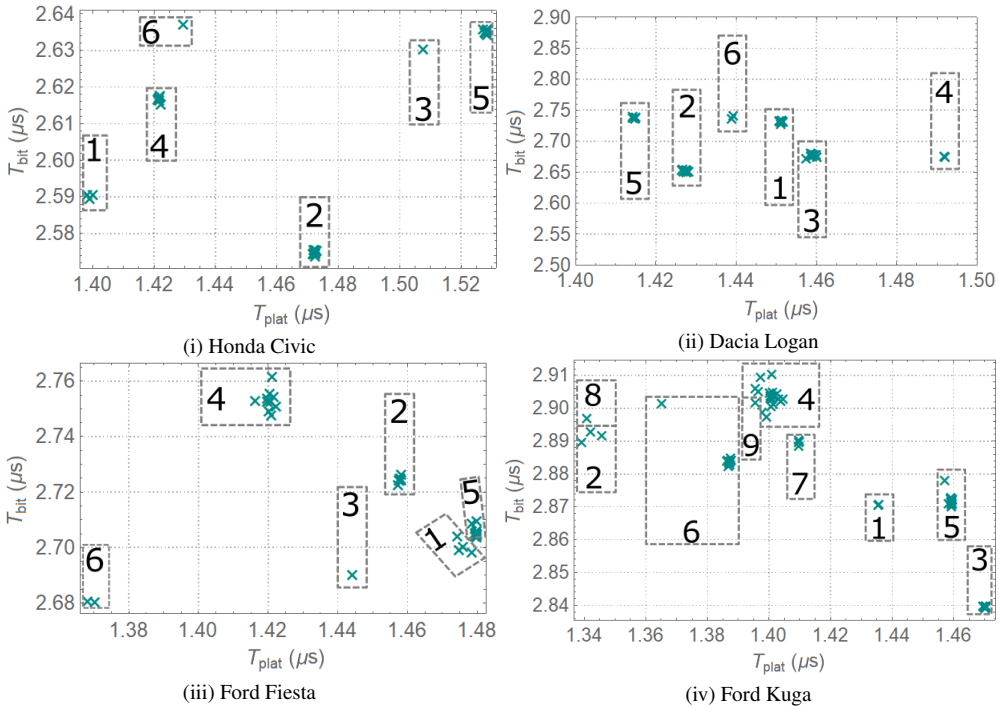
mean-max voltage separation presented in Figure 5.7 (i) shows the clusters for all ECUs with the mean and maximum voltage in the range of $1.8V$ to $2.3V$. Since the clock skew separates ECU_2 from ECU_3, the voltage patterns for bits that are part of frames with the IDs $18E$ (ECU_3) and 091 (ECU_4) are shown in Figure 5.9 (i) and (ii), for comparison purposes. Considering that voltage patterns are different for these bits and the separation using bit and plateau time from Figure 5.8 (i) show 6 separate clusters with values between $1.38\mu s$ and $1.54\mu s$ for the plateau time and $2.56\mu s$ to $2.64\mu s$ for the bit time, it confirms the findings of 6 ECUs. The physical characteristic values determined for all IDs from the Honda Civic vehicle and the grouping as part of ECUs are shown in Table 5.3.

In the **Opel Corsa** passenger vehicle there are 4 ECUs found based on 29 cyclic messages transmitted on the CAN bus. For this vehicle, both the clock skews and the voltage features allow a clear separation of transmitters as distinct ECUs. The mean and maximum voltage separation is shown in Figure 5.7 (ii) where 4 clusters with mean and maximum voltage between $1.9V$ and $2.5V$ can be easily distinguished. The physical characteristic values determined for all IDs from the Opel Corsa vehicle and the grouping as ECUs based on the frame identifiers are shown in Table 5.4.

In the **Hyundai i20** passenger vehicle there are 7 ECUs found based on 40 cyclic messages transmitted on the CAN bus. Similar to Honda Civic and Opel Corsa, the

Table 5.3: Physical characteristics for Honda Civic

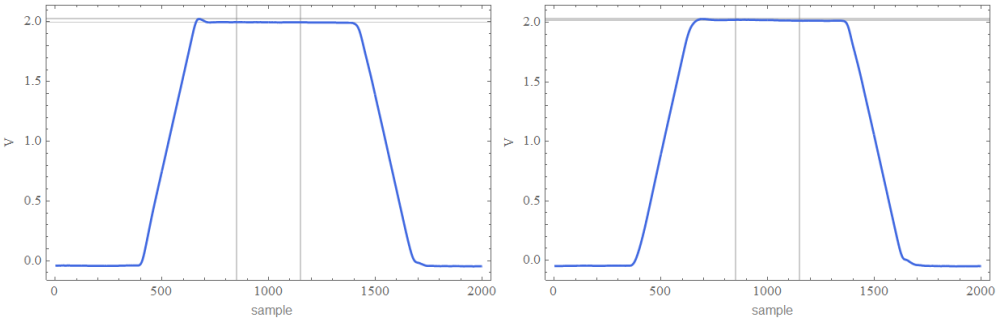
No.	ECU	ID	Cycle	$\mathbb{C}_{\mathrm{skew}}$	$\mathbb{V}_{\mathrm{mean}}$	$\mathbb{V}_{\mathrm{max}}$	$\mathbb{T}_{\mathrm{bit}}$	$\mathbb{T}_{\mathrm{plat}}$
1	ECU_1	039	40	1.000220	1.871	1.870	2.591	1.400
2	ECU_1	305	100	1.000220	1.884	1.883	2.590	1.399
3	ECU_1	401	300	1.000220	1.882	1.881	2.591	1.399
4	ECU_2	1A6	20	1.000240	1.997	2.017	2.575	1.473
5	ECU_2	21E	40	1.000240	1.996	2.0167	2.575	1.473
6	ECU_2	221	40	1.000240	1.994	2.014	2.575	1.473
7	ECU_2	294	40	1.000240	1.997	2.018	2.576	1.473
8	ECU_2	295	40	1.000240	1.996	2.015	2.576	1.473
9	ECU_2	309	100	1.000240	2.001	2.020	2.575	1.472
10	ECU_2	372	100	1.000240	1.996	2.016	2.574	1.473
11	ECU_2	374	100	1.000240	1.997	2.018	2.575	1.473
12	ECU_2	377	100	1.000240	1.994	2.013	2.575	1.472
13	ECU_2	378	100	1.000240	1.995	2.015	2.575	1.473
14	ECU_2	386	100	1.000240	1.994	2.014	2.576	1.473
15	ECU_2	405	300	1.000240	1.991	2.011	2.576	1.472
16	ECU_2	428	300	1.000240	1.987	2.004	2.575	1.472
17	ECU_2	42D	300	1.000240	1.991	2.011	2.575	1.472
18	ECU_2	42E	300	1.000240	1.993	2.012	2.576	1.473
19	ECU_3	18E	10	0.999994	2.003	2.029	2.631	1.508
20	ECU_4	091	10	0.999969	2.018	2.027	2.617	1.422
21	ECU_4	19B	10	0.999968	2.019	2.028	2.617	1.422
22	ECU_4	1A4	20	0.999968	2.018	2.028	2.617	1.422
23	ECU_4	1AA	20	0.999968	2.016	2.026	2.617	1.422
24	ECU_4	1B0	20	0.999968	2.020	2.029	2.617	1.422
25	ECU_4	1D0	20	0.999968	2.020	2.030	2.617	1.422
26	ECU_4	1EA	20	0.999968	2.019	2.028	2.617	1.421
27	ECU_4	255	40	0.999968	2.018	2.027	2.618	1.422
28	ECU_4	3D9	200	0.9999668	2.018	2.028	2.616	1.422
29	ECU_4	406	300	0.999965	2.017	2.027	2.618	1.422
30	ECU_5	13C	10	0.999860	2.107	2.155	2.635	1.528
31	ECU_5	158	10	0.999860	2.108	2.155	2.635	1.528
32	ECU_5	17C	10	0.999860	2.107	2.154	2.636	1.528
33	ECU_5	1DC	20	0.999861	2.105	2.153	2.635	1.528
34	ECU_5	1ED	20	0.999861	2.103	2.151	2.635	1.529
35	ECU_5	320	100	0.999861	2.105	2.152	2.636	1.528
36	ECU_5	324	100	0.999861	2.105	2.153	2.635	1.528
37	ECU_5	328	100	0.999861	2.107	2.155	2.636	1.528
38	ECU_5	3D7	200	0.999862	2.109	2.157	2.636	1.529
39	ECU_5	400	300	0.999861	2.107	2.155	2.636	1.529
40	ECU_5	40C	300	0.999861	2.105	2.153	2.635	1.529
41	ECU_5	454	300	0.999860	2.105	2.154	2.636	1.528
42	ECU_5	465	300	0.999860	2.105	2.152	2.635	1.528
43	ECU_6	156	10	1.000030	2.194	2.204	2.637	1.430



(i) Honda Civic (ii) Dacia Logan

(iii) Ford Fiesta (iv) Ford Kuga

Figure 5.8: Bit-plateau time separation



(i) ID 18E (ii) ID 091

Figure 5.9: A zero bit from ID 18E and one from ID 091 for Honda Civic [31]

separation of ECUs was clear based on clock skews and voltage features. The mean and maximum voltage separation is shown in Figure 5.7 (iii) where all 7 groups of IDs can be identified in the $1.9V - 2.5V$ range for both voltage features. The physical characteristic values determined for all IDs from the Hyundai i20 vehicle and the grouping as part of ECUs are shown in Table 5.5.

In the **Dacia Duster** passenger vehicle there are 3 ECUs found based on 12 cyclic

Table 5.4: Physical characteristics for Opel Corsa

No.	ECU	ID	Cycle	$\mathbb{C}_{\mathrm{skew}}$	$\mathbb{V}_{\mathrm{mean}}$	$\mathbb{V}_{\mathrm{max}}$	$\mathbb{T}_{\mathrm{bit}}$	$\mathbb{T}_{\mathrm{plat}}$
1	ECU_1	361	100	0.997619	2.443	2.466	2.582	1.447
2	ECU_1	460	100	0.997619	2.438	2.462	2.582	1.447
3	ECU_1	1F1	100	0.997619	2.441	2.465	2.582	1.446
4	ECU_1	1E1	30	0.997618	2.439	2.463	2.582	1.447
5	ECU_1	0F1	10	0.997619	2.441	2.465	2.582	1.447
6	ECU_1	3F1	250	0.997617	2.439	2.463	2.581	1.448
7	ECU_1	440	1,000	0.997619	2.440	2.464	2.582	1.447
8	ECU_2	265	1,000	0.999940	2.027	2.031	2.666	1.520
9	ECU_2	2F9	50	0.999940	2.025	2.032	2.666	1.522
10	ECU_2	1C9	20	0.999939	2.022	2.030	2.666	1.522
11	ECU_2	1E9	20	0.999939	2.020	2.029	2.666	1.523
12	ECU_2	0C1	10	0.999939	2.024	2.031	2.666	1.522
13	ECU_2	363	100	0.999940	2.023	2.030	2.667	1.522
14	ECU_2	0C5	10	0.999939	2.024	2.031	2.666	1.522
15	ECU_2	530	1,000	0.999940	2.027	2.031	2.670	1.521
16	ECU_3	370	500	0.999998	1.946	1.949	2.606	1.378
17	ECU_3	1E5	10	0.999998	1.951	1.951	2.605	1.381
18	ECU_4	3F9	250	1.000060	2.200	2.269	2.612	1.508
19	ECU_4	772	1,000	1.000060	2.196	2.263	2.613	1.507
20	ECU_4	4C1	500	1.000060	2.195	2.262	2.616	1.509
21	ECU_4	4D1	500	1.000060	2.196	2.262	2.614	1.509
22	ECU_4	3E9	100	1.000060	2.198	2.266	2.612	1.508
23	ECU_4	3D1	100	1.000060	2.198	2.264	2.612	1.508
24	ECU_4	2C5	50	1.000060	2.199	2.266	2.614	1.508
25	ECU_4	1BD	50	1.000060	2.199	2.265	2.613	1.508
26	ECU_4	1BC	10	1.000060	2.198	2.264	2.611	1.507
27	ECU_4	0C9	10	1.000060	2.199	2.266	2.613	1.508
28	ECU_4	1C1	20	1.000060	2.199	2.266	2.613	1.508
29	ECU_4	1F5	20	1.000060	2.201	2.267	2.612	1.508

Table 5.5: Physical characteristics for Hyundai i20

No.	ECU	ID	Cycle	$\mathbb{C}_{\mathrm{skew}}$	$\mathbb{V}_{\mathrm{mean}}$	$\mathbb{V}_{\mathrm{max}}$	$\mathbb{T}_{\mathrm{bit}}$	$\mathbb{T}_{\mathrm{plat}}$
1	ECU_1	593	200	0.999477	1.950	1.949	2.718	1.374
2	ECU_2	043	1,000	0.999163	1.966	1.965	2.693	1.351
3	ECU_2	044	1,000	0.999163	1.968	1.963	2.691	1.355
4	ECU_2	383	20	0.999163	1.970	1.968	2.706	1.357
5	ECU_3	2B0	10	1.001009	1.987	1.996	2.678	1.399
6	ECU_3	381	20	1.001009	1.991	2.000	2.679	1.398
7	ECU_3	251	10	1.001009	1.991	2.001	2.680	1.397
8	ECU_4	549	100	1.000030	2.160	2.173	2.731	1.447
9	ECU_4	5CE	100	1.000030	2.157	2.172	2.729	1.449
10	ECU_4	5CF	100	1.000032	2.158	2.171	2.731	1.446
11	ECU_4	547	100	1.000036	2.160	2.173	2.732	1.446
12	ECU_4	1BF	10	1.000031	2.156	2.172	2.730	1.447
13	ECU_4	316	10	1.000031	2.155	2.169	2.729	1.448
14	ECU_4	18F	10	1.000031	2.159	2.173	2.729	1.448
15	ECU_4	260	10	1.000032	2.158	2.171	2.730	1.447
16	ECU_4	329	10	1.000033	2.159	2.173	2.730	1.447
17	ECU_4	4E5	100	1.000031	2.156	2.169	2.733	1.447
18	ECU_4	4E6	100	1.000031	2.160	2.174	2.732	1.447
19	ECU_4	545	100	1.000036	2.158	2.172	2.728	1.448
20	ECU_4	4E7	100	1.000031	2.158	2.171	2.731	1.448
21	ECU_4	200	10	1.000032	2.158	2.172	2.728	1.447
22	ECU_4	492	50	1.000031	2.160	2.172	2.731	1.446
23	ECU_4	556	100	1.000031	2.156	2.171	2.727	1.450
24	ECU_4	557	100	1.000033	2.156	2.170	2.727	1.448
25	ECU_5	164	10	0.999958	2.199	2.210	2.710	1.472
26	ECU_5	220	10	0.999958	2.199	2.210	2.710	1.471
27	ECU_5	153	10	0.999958	2.198	2.210	2.706	1.472
28	ECU_5	387	20	0.999958	2.199	2.209	2.713	1.471
29	ECU_5	386	20	0.999958	2.199	2.211	2.709	1.472
30	ECU_5	507	100	0.999958	2.199	2.211	2.707	1.472
31	ECU_6	500	100	0.999542	2.294	2.297	2.800	1.419
32	ECU_6	5A0	1,000	0.999543	2.298	2.300	2.827	1.422
33	ECU_6	5A1	1,000	0.999543	2.298	2.299	2.804	1.418
34	ECU_7	4F1	20	0.999936	2.392	2.395	2.669	1.411
35	ECU_7	50C	100	0.999936	2.393	2.398	2.671	1.412
36	ECU_7	50E	200	0.999936	2.389	2.395	2.671	1.413
37	ECU_7	541	100	0.999937	2.393	2.397	2.668	1.411
38	ECU_7	52A	200	0.999937	2.396	2.400	2.668	1.411
39	ECU_7	553	200	0.999937	2.389	2.396	2.669	1.413
40	ECU_7	5B0	1,000	0.999936	2.390	2.395	2.667	1.412

Table 5.6: Physical characteristics for Dacia Duster

No.	ECU	ID	Cycle	$\mathbb{C}_{\text{skew}}$	$\mathbb{V}_{\text{mean}}$	$\mathbb{V}_{\text{max}}$	$\mathbb{T}_{\text{bit}}$	$\mathbb{T}_{\text{plat}}$
1	ECU_1	161	10	0.999967	2.464	2.535	2.477	1.452
2	ECU_1	181	10	0.999967	2.464	2.535	2.477	1.452
3	ECU_1	1F9	10	0.999967	2.463	2.534	2.477	1.452
4	ECU_1	511	100	0.999967	2.462	2.527	2.478	1.450
5	ECU_1	65C	100	0.999967	2.464	2.533	2.477	1.451
6	ECU_1	5DD	100	0.999967	2.463	2.534	2.477	1.452
7	ECU_1	551	100	0.999967	2.462	2.532	2.477	1.452
8	ECU_2	284	20	0.999975	1.932	1.952	2.555	1.526
9	ECU_2	285	20	0.999975	1.936	1.959	2.554	1.530
10	ECU_2	244	20	0.999975	1.935	1.956	2.555	1.530
11	ECU_2	354	40	0.999975	1.936	1.959	2.555	1.531
12	ECU_3	1A5	10	1.000220	2.084	2.084	2.547	1.404

messages transmitted on the CAN bus. This is the smallest number of ECUs and frames from the dataset, for a single vehicle. Same as for the previous vehicles, the ECU separation was clear both on clock skews and voltage features. The mean and maximum voltage separation is shown in Figure 5.7 (iv) with 3 clusters that are distinguishable between $1.9V$ and $2.6V$. The physical characteristic values determined for all IDs from the Dacia Duster vehicle and the grouping as part of ECUs are shown in Table 5.6.

In the **Dacia Logan** passenger vehicle there are 6 ECUs found based on 46 cyclic messages transmitted on the CAN bus. Since the ECUs from Dacia Duster are clearly separated, the initial assumption was that separation would be clear for this vehicle as well, but this is not the case. That is because two ECUs, ECU_2 and ECU_3, which are clearly separated based on voltage fingerprints, have a difference of the determined clock skew of only $1ppm$ that is of 0.000001. The mean and maximum voltage separation is shown in Figure 5.7 (v) with 6 clusters that are distinguishable between $1.9V$ and $2.3V$. The voltage separation using bit and plateau time from Figure 5.8 (ii) shows 6 clusters that correspond to 6 distinct ECUs connected on the OBD-II CAN bus of the Dacia Logan. The physical characteristic values determined for all IDs from the Dacia Logan vehicle and the grouping as part of ECUs are shown in Table 5.7.

In the **Hyundai ix35** passenger vehicle there are 6 ECUs found based on 26 cyclic messages transmitted on the CAN bus. The ECU separation is clear both on clock skews and voltage features. The mean and maximum voltage separation is shown in Figure 91 5.7 (vi) with 6 clusters that are distinguishable between $1.9V$ and $2.3V$. The physical characteristic values determined for all IDs from the Hyundai ix35 vehicle and the grouping as part of ECUs are shown in Table 5.8.

In the **Ford Fiesta** passenger vehicle there are 6 ECUs found based on 46 cyclic messages transmitted on the CAN bus. The ECU separation was problematic on clock skews since the observed timings on some IDs were not stable over time. The exemplification of changes with respect to timing is shown in Figure 5.10. For ID 073 (i) the cycle time is the same over all 6000 frames, so the obtained clock skew remains the same. On the other hand, for IDs 364 (ii) and 360 (ii) there is a change in the offset value over time which is unexpected. This means that clock skew is different based on the time slot used to compute it. Therefore, the clock skew obtained from the first half of frames for IDs

Table 5.7: Physical characteristics for Dacia Logan

No.	ECU	ID	Cycle	$\mathbb{C}_{\mathrm{skew}}$	$\mathbb{V}_{\mathrm{mean}}$	$\mathbb{V}_{\mathrm{max}}$	$\mathbb{T}_{\mathrm{bit}}$	$\mathbb{T}_{\mathrm{plat}}$
1	ECU_1	500	100	0.997246	1.940	1.941	2.734	1.451
2	ECU_1	1B0	20	0.999574	1.940	1.941	2.734	1.452
3	ECU_1	552	100	0.999574	1.940	1.941	2.735	1.452
4	ECU_1	657	100	0.999574	1.941	1.942	2.734	1.452
5	ECU_1	2BC	100	0.999574	1.940	1.941	2.734	1.451
6	ECU_1	69F	1,000	0.999574	1.941	1.941	2.729	1.451
7	ECU_1	4DE	100	0.999574	1.942	1.940	2.736	1.448
8	ECU_1	55D	100	0.999574	1.940	1.940	2.735	1.451
9	ECU_1	5DE	100	0.999574	1.940	1.941	2.733	1.452
10	ECU_1	575	100	0.999574	1.940	1.941	2.734	1.451
11	ECU_1	45C	100	0.999574	1.940	1.942	2.733	1.451
12	ECU_1	5DF	100	0.999574	1.941	1.942	2.734	1.451
13	ECU_1	350	100	0.999574	1.940	1.941	2.735	1.452
14	ECU_1	4AC	100	0.999574	1.941	1.941	2.734	1.451
15	ECU_2	217	20	0.999974	2.046	2.050	2.655	1.428
16	ECU_2	2C6	20	0.999974	2.044	2.048	2.655	1.428
17	ECU_2	2A9	20	0.999974	2.044	2.049	2.654	1.427
18	ECU_2	18A	10	0.999974	2.045	2.050	2.655	1.428
19	ECU_2	186	10	0.999974	2.044	2.048	2.654	1.428
20	ECU_2	66A	100	0.999974	2.045	2.048	2.655	1.427
21	ECU_2	511	100	0.999974	2.043	2.046	2.652	1.428
22	ECU_2	1F6	10	0.999974	2.045	2.049	2.655	1.428
23	ECU_2	5DA	100	0.999974	2.043	2.046	2.653	1.428
24	ECU_2	648	100	0.999974	2.043	2.046	2.653	1.428
25	ECU_2	65C	100	0.999974	2.042	2.045	2.653	1.428
26	ECU_2	41A	100	0.999974	2.044	2.047	2.652	1.427
27	ECU_2	41D	100	0.999974	2.046	2.049	2.657	1.427
28	ECU_3	090	10	0.999973	2.118	2.128	2.679	1.459
29	ECU_3	0C6	10	0.999973	2.116	2.126	2.681	1.459
30	ECU_3	666	100	0.999973	2.124	2.133	2.674	1.458
31	ECU_3	352	40	0.999973	2.117	2.128	2.677	1.460
32	ECU_3	29C	20	0.999973	2.119	2.129	2.678	1.459
33	ECU_3	12E	10	0.999973	2.117	2.128	2.680	1.459
34	ECU_3	242	20	0.999973	2.116	2.127	2.680	1.460
35	ECU_3	354	40	0.999973	2.122	2.133	2.678	1.459
36	ECU_3	2B7	20	0.999973	2.118	2.129	2.680	1.459
37	ECU_3	29A	20	0.999973	2.118	2.128	2.679	1.460
38	ECU_3	5D7	100	0.999973	2.118	2.128	2.682	1.459
39	ECU_4	1A0	100	1.000530	2.190	2.222	2.676	1.492
40	ECU_4	62B	100	1.000530	2.192	2.225	2.677	1.492
41	ECU_5	4F8	100	0.999507	2.201	2.222	2.739	1.415
42	ECU_5	646	500	0.999507	2.200	2.222	2.742	1.414
43	ECU_5	3B7	100	0.999507	2.199	2.220	2.738	1.415
44	ECU_5	6FB	3,000	0.999507	2.200	2.220	2.740	1.415
45	ECU_6	564	100	1.000510	2.221	2.237	2.739	1.439
46	ECU_6	653	100	1.000510	2.229	2.246	2.743	1.439

Table 5.8: Physical characteristics for Hyundai ix35

No.	ECU	ID	Cycle	$\mathcal{C}_{\text{skew}}$	$\mathbb{V}_{\text{mean}}$	$\mathbb{V}_{\text{max}}$	$\mathbb{T}_{\text{bit}}$	$\mathbb{T}_{\text{plat}}$
1	ECU_1	350	10	0.999534	1.937	1.934	2.664	1.332
2	ECU_2	5E4	100	0.999966	2.115	2.130	2.744	1.532
3	ECU_2	165	10	0.999966	2.120	2.132	2.747	1.527
4	ECU_2	2B0	10	0.999966	2.112	2.127	2.746	1.535
5	ECU_3	4F0	20	0.998440	2.136	2.147	2.667	1.415
6	ECU_3	690	100	0.998440	2.138	2.148	2.669	1.409
7	ECU_4	430	21	1.000070	2.137	2.165	2.639	1.455
8	ECU_4	4B1	21	1.000070	2.133	2.162	2.638	1.456
9	ECU_4	4D0	21	1.000070	2.133	2.161	2.638	1.454
10	ECU_4	153	7	1.000070	2.122	2.154	2.637	1.456
11	ECU_4	164	7	1.000070	2.137	2.165	2.637	1.456
12	ECU_4	220	7	1.000070	2.134	2.162	2.637	1.455
13	ECU_4	1F1	21	1.000070	2.138	2.165	2.637	1.456
14	ECU_5	316	10	1.000010	2.165	2.213	2.673	1.480
15	ECU_5	0A1	10	1.000010	2.165	2.212	2.673	1.481
16	ECU_5	0A0	10	1.000010	2.164	2.212	2.673	1.481
17	ECU_5	18F	10	1.000010	2.166	2.213	2.673	1.481
18	ECU_5	329	10	1.000010	2.165	2.213	2.673	1.481
19	ECU_5	260	10	1.000010	2.166	2.213	2.673	1.481
20	ECU_5	2A0	10	1.000010	2.165	2.214	2.673	1.481
21	ECU_5	545	10	1.000020	2.165	2.213	2.673	1.480
22	ECU_6	429	20	0.999900	2.198	2.205	2.666	1.428
23	ECU_6	428	20	0.999900	2.191	2.195	2.640	1.417
24	ECU_6	5A0	1,000	0.999921	2.201	2.212	2.674	1.445
25	ECU_6	5A2	1,000	0.999921	2.204	2.213	2.673	1.443
26	ECU_6	5A1	1,005	0.999920	2.197	2.206	2.648	1.442

364 and 360 is different than the clock skew computed from the second half of frames. This means that, by selecting a low number of frames, it leads to an unreliable separation of ECUs using estimated clock skews. Nevertheless, the ECU separation is clear if voltage features are used, as shown in Figure 5.7 (vii) and Figure 5.8 (iii). In both figures, the 6 clusters that correspond to 6 distinct ECUs can be identified. The physical characteristic values determined for all IDs from the Ford Fiesta vehicle and the grouping as part of ECUs are shown in Table 5.9. Some IDs were omitted from the analysis and the experimental results since they are on-event (non-cyclic), i.e., 455, 720, 727, 728, 72F, 7A5 and 7AD.

In the **Ford Kuga** passenger vehicle there are 9 ECUs found based on 70 cyclic messages transmitted on the CAN bus. This is the highest number of ECUs from the dataset, for a single vehicle. In the dataset for Ford Kuga, there are frames with the same voltage fingerprint but with different clock skew, which suggests that some frames are gatewayed by the nodes from other vehicle buses. An example of this problem is shown on the top side of Figure 5.11 with two IDs that originate from ECU_6, i.e., 1D0 (i) and 208 (ii). In case of the ID number 1D0, the measured cycle time variation is of $\pm 4ms$ from the expected $20ms$ cycle time, while for ID 208 the measured cycle time variation is of $\pm 7ms$ from the expected $25ms$ cycle time. Since the second ID is quite noisy, the assumption is that ID 208 is the one re-transmitted by the gateway ECU from another in-vehicle bus. The clock skew is of 0.999956 for ID 1D0 and of 1.002190 for ID 208 with the clock skew accumulation (iii) shown on the top side of Figure 5.11. Even though multiple clock skews are determined, the ECU separation is clear if voltage features are used, as shown in Figure 5.7 (viii) and Figure 5.8 (iv). In both Figures there are 9 clusters visible that correspond to 9 distinct ECUs. The physical characteristic values determined

Table 5.9: Physical characteristics for Ford Fiesta

No.	ECU	ID	Cycle	$\mathbb{C}_{\mathrm{skew}}$	$\mathbb{V}_{\mathrm{mean}}$	$\mathbb{V}_{\mathrm{max}}$	$\mathbb{T}_{\mathrm{bit}}$	$\mathbb{T}_{\mathrm{plat}}$
1	ECU_1	023	100	0.999861	2.117	2.158	2.705	1.475
2	ECU_1	04A	100	0.999861	2.113	2.154	2.701	1.476
3	ECU_1	04B	100	0.999861	2.113	2.152	2.699	1.475
4	ECU_1	460	100	0.999862	2.108	2.148	2.699	1.479
5	ECU_2	073	10	0.999948	2.154	2.173	2.725	1.458
6	ECU_2	090	10	0.999948	2.151	2.168	2.725	1.458
7	ECU_2	20E	10	0.999948	2.155	2.173	2.725	1.458
8	ECU_2	20F	10	0.999948	2.155	2.174	2.725	1.458
9	ECU_2	211	10	0.999948	2.152	2.169	2.725	1.458
10	ECU_2	212	100	0.999946	2.150	2.167	2.723	1.457
11	ECU_2	213	20	0.999948	2.156	2.175	2.725	1.458
12	ECU_2	215	20	0.999946	2.150	2.167	2.725	1.458
13	ECU_2	216	20	0.999946	2.150	2.168	2.727	1.458
14	ECU_2	2C3	1,000	0.999946	2.160	2.180	2.726	1.458
15	ECU_2	4B0	10	0.999948	2.155	2.175	2.725	1.458
16	ECU_3	150	25	1.000000	2.182	2.200	2.691	1.444
17	ECU_4	190	20	1.002000	2.212	2.234	2.753	1.421
18	ECU_4	275	100	1.001990	2.214	2.236	2.755	1.422
19	ECU_4	400	100	1.001990	2.214	2.236	2.749	1.420
20	ECU_4	405	100	1.002000	2.208	2.228	2.753	1.417
21	ECU_4	430	100	1.002000	2.212	2.234	2.756	1.421
22	ECU_4	432	100	1.002000	2.218	2.240	2.762	1.421
23	ECU_4	433	100	1.001990	2.212	2.235	2.751	1.423
24	ECU_4	4E3	30	1.002000	2.210	2.232	2.754	1.420
25	ECU_4	2C1	1,000	1.001990	2.211	2.233	2.753	1.420
26	ECU_4	4F2	1,000	1.001990	2.209	2.229	2.748	1.421
27	ECU_5	0FD	20	0.999908	2.242	2.312	2.705	1.480
28	ECU_5	200	10	0.999908	2.243	2.312	2.705	1.480
29	ECU_5	201	10	0.999908	2.241	2.311	2.705	1.480
30	ECU_5	203	30	0.999908	2.245	2.314	2.706	1.480
31	ECU_5	205	10	0.999908	2.241	2.311	2.705	1.480
32	ECU_5	228	25	0.999908	2.242	2.312	2.705	1.480
33	ECU_5	231	10	0.999908	2.238	2.308	2.704	1.480
34	ECU_5	232	10	0.999908	2.243	2.313	2.705	1.480
35	ECU_5	261	50	0.999908	2.246	2.315	2.709	1.479
36	ECU_5	268	10	0.999908	2.242	2.311	2.705	1.480
37	ECU_5	280	50	0.999908	2.243	2.312	2.706	1.479
38	ECU_5	2BA	100	0.999906	2.244	2.313	2.706	1.480
39	ECU_5	360	10	0.999908	2.243	2.313	2.705	1.480
40	ECU_5	364	30	0.999908	2.242	2.312	2.705	1.480
41	ECU_5	420	100	0.999909	2.242	2.312	2.705	1.480
42	ECU_5	424	100	0.999910	2.243	2.313	2.706	1.480
43	ECU_5	428	100	0.999906	2.243	2.312	2.705	1.480
44	ECU_5	4F1	1,000	0.999905	2.240	2.311	2.710	1.480
45	ECU_6	080	15	0.956060	2.433	2.450	2.681	1.370
46	ECU_6	240	10	0.956059	2.433	2.449	2.681	1.368

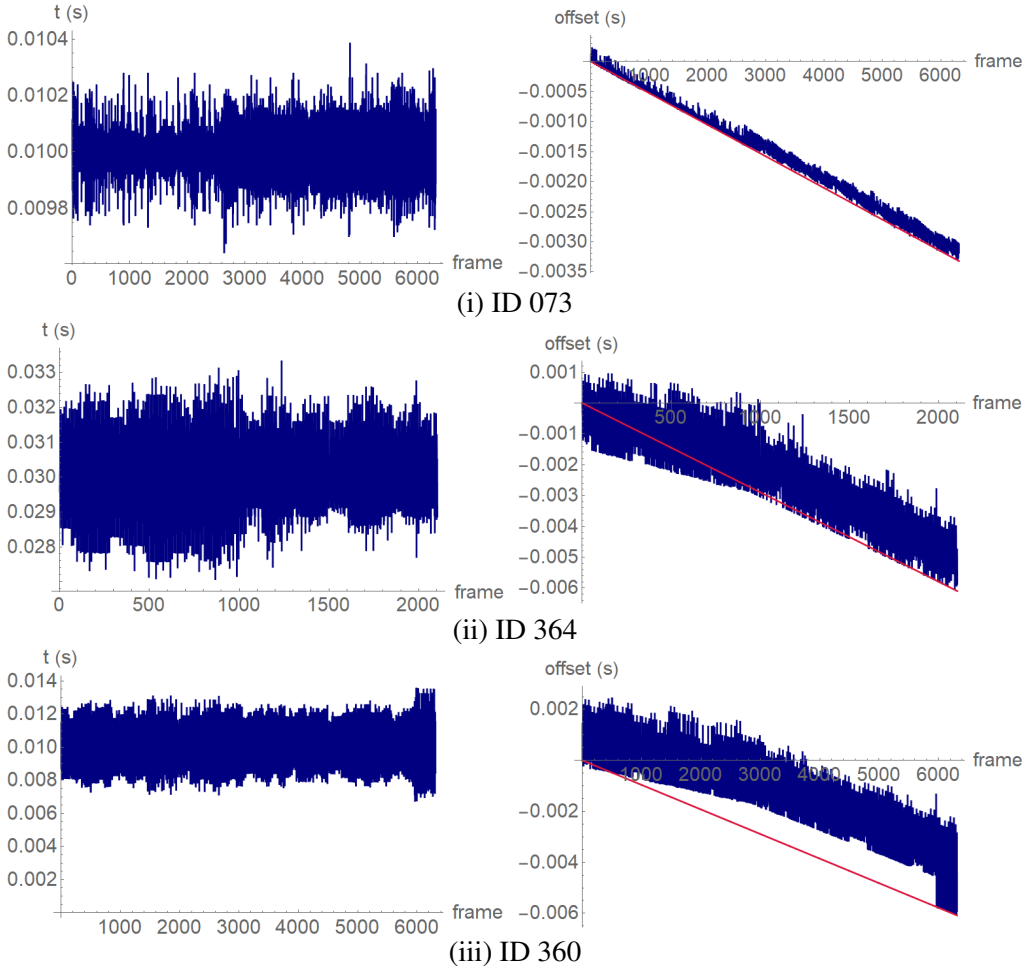


Figure 5.10: Cycle time (left) and offset (right) for 3 IDs in the Ford Fiesta

for all IDs from the Ford Kuga vehicle and the grouping as part of ECUs are shown in Table 5.10. One ID was omitted from the analysis and the experimental results since it is on-event (non-cyclic), i.e., 35E.

In the **Ford Ecosport** passenger vehicle there are 4 ECUs found based on 87 cyclic messages transmitted on the CAN bus. This is the highest number of IDs from the dataset, for a single vehicle. Similar problems regarding different clock skews determined for the same ECUs are present for Ford Ecosport as there are for Ford Kuga. An example of this problem is shown on the bottom side of Figure 5.11 with two IDs that originate from ECU_4, i.e., 049 (iv) and 091 (v). Their cycle time is of $20ms$. The reception time for both IDs is noisy, so it is unknown if one of them or both are transmitted by the node which acts as a gateway ECU. The clock skew is of 0.999926 for ID 049 and of 0.999611 for

Table 5.10: Physical characteristics for Ford Kuga

No.	ECU	ID	Cycle	$\mathbb{C}_{\text{skew}}$	$\mathbb{V}_{\text{mean}}$	$\mathbb{V}_{\text{max}}$	$\mathbb{T}_{\text{bit}}$	$\mathbb{T}_{\text{plat}}$
1	ECU_1	140	20	0.999661	1.943	1.968	2.871	1.436
2	ECU_1	0B0	20	0.999661	1.940	1.965	2.871	1.436
3	ECU_2	455	100	0.999995	2.115	2.127	2.893	1.342
4	ECU_2	3EA	1,000	0.999995	2.115	2.127	2.892	1.346
5	ECU_2	3E2	1,000	0.999995	2.121	2.131	2.890	1.339
6	ECU_3	2B0	40	1.000010	2.104	2.183	2.840	1.470
7	ECU_3	06A	20	1.000010	2.105	2.186	2.840	1.471
8	ECU_3	050	10	1.000010	2.106	2.187	2.840	1.471
9	ECU_3	0E0	20	1.000010	2.106	2.186	2.839	1.470
10	ECU_3	0D0	20	1.000010	2.105	2.185	2.840	1.470
11	ECU_3	0F0	10	1.000010	2.104	2.185	2.840	1.470
12	ECU_3	0F5	10	1.000010	2.106	2.184	2.840	1.470
13	ECU_3	100	20	1.000010	2.106	2.187	2.840	1.470
14	ECU_4	435	300	0.999983	2.134	2.178	2.905	1.400
15	ECU_4	40A	125	0.999983	2.134	2.178	2.903	1.403
16	ECU_4	581	1,000	0.999980	2.130	2.175	2.905	1.401
17	ECU_4	360	150	0.999982	2.135	2.178	2.901	1.401
18	ECU_4	310	100	0.999983	2.136	2.179	2.903	1.401
19	ECU_4	260	25	0.999982	2.135	2.178	2.904	1.402
20	ECU_4	150	20	0.999982	2.135	2.179	2.903	1.401
21	ECU_4	0C8	20	0.999982	2.134	2.177	2.904	1.401
22	ECU_4	030	10	0.999983	2.134	2.179	2.903	1.405
23	ECU_4	17E	100	0.999983	2.135	2.179	2.902	1.404
24	ECU_4	290	30	0.999983	2.136	2.180	2.904	1.401
25	ECU_4	400	250	0.999983	2.135	2.178	2.905	1.403
26	ECU_4	380	300	0.999984	2.137	2.181	2.902	1.402
27	ECU_4	3B4	300	0.999984	2.135	2.178	2.898	1.400
28	ECU_4	420	600	0.999984	2.135	2.178	2.899	1.398
29	ECU_4	405	250	1.028590	2.135	2.178	2.903	1.401
30	ECU_5	090	10	1.000010	2.134	2.208	2.872	1.460
31	ECU_5	060	15	1.000010	2.135	2.211	2.873	1.459
32	ECU_5	2F0	90	1.000010	2.135	2.207	2.871	1.458
33	ECU_5	280	30	1.000010	2.134	2.207	2.872	1.459
34	ECU_5	200	25	1.000010	2.133	2.207	2.873	1.459
35	ECU_5	270	30	1.000010	2.134	2.208	2.873	1.459
36	ECU_5	0A0	15	1.000010	2.134	2.207	2.871	1.460
37	ECU_5	1A0	20	1.000010	2.133	2.207	2.872	1.460
38	ECU_5	1B0	30	1.000010	2.134	2.208	2.873	1.459
39	ECU_5	130	20	1.000010	2.133	2.207	2.873	1.460
40	ECU_5	138	20	1.000010	2.132	2.206	2.871	1.460
41	ECU_5	080	20	1.000010	2.132	2.206	2.871	1.460
42	ECU_5	120	20	1.000010	2.134	2.208	2.873	1.460
43	ECU_5	070	20	1.000010	2.133	2.207	2.870	1.460
44	ECU_5	0C0	20	1.000010	2.133	2.208	2.872	1.459
45	ECU_5	0F8	20	1.000010	2.134	2.208	2.871	1.460
46	ECU_5	2D8	60	1.000010	2.136	2.210	2.873	1.459
47	ECU_5	340	120	1.000020	2.140	2.212	2.878	1.457
48	ECU_6	2D0	40	0.999957	2.158	2.170	2.885	1.388
49	ECU_6	218	30	0.999956	2.160	2.172	2.884	1.387
50	ECU_6	252	20	0.999957	2.158	2.170	2.884	1.387
51	ECU_6	190	10	0.999956	2.155	2.167	2.884	1.387
52	ECU_6	2D4	60	0.999956	2.154	2.165	2.884	1.387
53	ECU_6	180	20	0.999956	2.155	2.167	2.884	1.387
54	ECU_6	1C0	20	0.999956	2.155	2.167	2.884	1.388
55	ECU_6	1D0	20	0.999956	2.154	2.166	2.883	1.388
56	ECU_6	1E0	20	0.999956	2.159	2.172	2.885	1.388
57	ECU_6	210	20	0.999956	2.158	2.170	2.884	1.387
58	ECU_6	160	20	0.999956	2.157	2.169	2.884	1.388
59	ECU_6	213	20	0.999956	2.153	2.165	2.883	1.387
60	ECU_6	388	801	1.000930	2.159	2.169	2.905	1.397
61	ECU_6	208	25	1.002190	2.156	2.169	2.911	1.401
62	ECU_6	2E0	70	1.000940	2.157	2.176	2.902	1.365
63	ECU_7	2A0	40	0.999600	2.163	2.180	2.891	1.410
64	ECU_7	2A5	40	0.999600	2.164	2.181	2.890	1.410
65	ECU_7	229	40	0.999600	2.167	2.185	2.890	1.410
66	ECU_7	170	20	0.999600	2.165	2.182	2.889	1.410
67	ECU_7	04A	1,000	1.002190	2.163	2.174	2.906	1.396
68	ECU_7	04B	1,000	1.002190	2.161	2.173	2.910	1.397
69	ECU_8	010	10	0.999997	2.181	2.193	2.897	1.341
70	ECU_9	269	30	1.000530	2.169	2.201	2.902	1.396

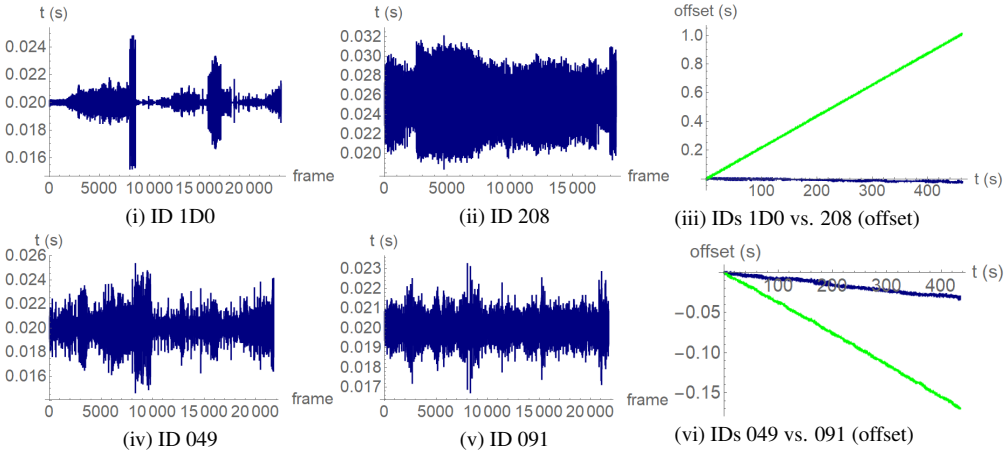


Figure 5.11: Cycle time and skews for IDs 1D0 and 208 in Ford Kuga and IDs 049 and 091 in Ford Ecosport

ID 091 with the clock skew accumulation (vi) shown on the bottom side of Figure 5.11. Even though multiple clock skews are determined for frames sent by ECU_1, ECU_3 or ECU_4, the ECU separation is clear if voltage features are used, as shown in Figure 5.7 (ix). By using the mean and maximum voltage separation, 4 clusters can be identified that correspond to 4 distinct ECUs. The physical characteristic values determined for all IDs from the Ford Ecosport vehicle and their grouping as part of ECUs are shown in Table 5.11.

5.5.2 Overview of intra-distances and inter-distances

Since the information regarding ECU classification is provided for each vehicle, the intra-distances and inter-distances are discussed in what follows. The intra-distances and inter-distances are shown as heatmaps in Figure 5.12 for clock skews and Figures 5.13 (i) to (iv) for separate voltage features. Then, all voltage features are used to measure the intra-distances and inter-distances in case mean voltage, maximum voltage, bit time and plateau time, so a better separation between nodes can be evaluated. This is shown in Figure 5.14. The thresholds used for the heatmaps are of $10ppm$ for separation of the skews, $25mV$ for separation of the mean and maximum voltage and $10ns$ for separation of the bit time and plateau time. The inter-distances are higher for ECUs from the same vehicle and, are sometimes very close, for ECUs in different vehicles.

There is a clear separation between most of the ECUs if clock skews are used, as shown in Figure 5.12. As mentioned earlier, in some vehicles, some ECUs with different voltage characteristics and the same or very similar clock skews were identified. This means that, for Ford Kuga and Ford Ecosport there are more groups of transmitters that can be classified using the clock skew compared to those using voltage features.

Table 5.11: Physical characteristics for Ford Ecosport

No.	ECU	ID	Cycle	$\mathcal{C}_{\text{skew}}$	$\mathbb{V}_{\text{mean}}$	$\mathbb{V}_{\text{max}}$	$\mathbb{T}_{\text{bit}}$	$\mathbb{T}_{\text{plat}}$
1	ECU_1	447	1,000	0.965410	1.906	1.916	2.647	1.417
2	ECU_1	041	20	0.982994	1.901	1.913	2.647	1.420
3	ECU_1	331	500	0.996618	1.899	1.911	2.648	1.420
4	ECU_1	3B3	500	0.978530	1.904	1.915	2.651	1.420
5	ECU_1	084	1,000	0.999990	1.905	1.917	2.643	1.419
6	ECU_1	3A9	20	0.997814	1.907	1.909	2.738	1.470
7	ECU_1	3A8	20	0.997814	1.906	1.908	2.718	1.471
8	ECU_1	3AB	200	0.997813	1.907	1.909	2.716	1.471
9	ECU_1	3AA	200	0.997813	1.906	1.908	2.717	1.472
10	ECU_1	40A	197	0.997962	1.903	1.915	2.648	1.421
11	ECU_1	3B7	250	0.982993	1.904	1.915	2.647	1.421
12	ECU_1	3B6	250	0.982993	1.901	1.913	2.646	1.420
13	ECU_1	3AE	1,000	0.997806	1.911	1.911	2.725	1.466
14	ECU_1	581	1,000	0.982989	1.901	1.913	2.648	1.421
15	ECU_1	3B4	1,000	0.982993	1.893	1.906	2.650	1.421
16	ECU_1	3E3	1,000	0.983002	1.903	1.915	2.646	1.421
17	ECU_1	3EB	1,000	0.983003	1.900	1.912	2.648	1.420
18	ECU_1	43C	1,000	0.983003	1.903	1.914	2.652	1.420
19	ECU_1	3B1	1,000	0.983001	1.904	1.917	2.654	1.422
20	ECU_1	3C7	1,000	0.982992	1.901	1.913	2.646	1.421
21	ECU_1	3C3	1,000	0.982993	1.903	1.914	2.649	1.420
22	ECU_1	38D	1,000	0.983001	1.909	1.918	2.654	1.418
23	ECU_1	3B8	500	0.983004	1.901	1.913	2.646	1.422
24	ECU_1	3B5	500	0.982994	1.902	1.914	2.645	1.421
25	ECU_1	42C	50	0.982992	1.899	1.912	2.647	1.421
26	ECU_1	242	39	1.008200	1.902	1.914	2.648	1.420
27	ECU_2	3E2	1,000	0.999998	2.116	2.131	2.668	1.405
28	ECU_2	3EA	1,000	0.999998	2.122	2.135	2.671	1.402
29	ECU_2	455	100	0.999998	2.114	2.127	2.670	1.404
30	ECU_3	43D	50	0.999999	2.196	2.234	2.682	1.478
31	ECU_3	43E	50	0.999996	2.196	2.235	2.684	1.478
32	ECU_3	42F	30	0.999998	2.196	2.235	2.684	1.478
33	ECU_3	171	30	1.000000	2.195	2.236	2.686	1.478
34	ECU_3	421	100	1.000000	2.197	2.236	2.684	1.478
35	ECU_3	424	100	1.000000	2.197	2.236	2.681	1.477
36	ECU_3	41F	100	1.000000	2.193	2.231	2.682	1.478
37	ECU_3	42D	100	1.000000	2.194	2.233	2.679	1.478
38	ECU_3	230	20	1.000000	2.193	2.232	2.682	1.479
39	ECU_3	595	1,000	1.000000	2.188	2.228	2.679	1.478
40	ECU_3	202	20	1.000000	2.194	2.233	2.684	1.478
41	ECU_3	179	100	1.000000	2.196	2.234	2.683	1.479
42	ECU_3	200	20	1.000000	2.194	2.233	2.683	1.478
43	ECU_3	178	100	1.000000	2.195	2.234	2.684	1.478
44	ECU_3	17C	100	1.000000	2.197	2.236	2.684	1.478
45	ECU_3	156	100	1.000000	2.196	2.234	2.683	1.478
46	ECU_3	166	100	1.000000	2.195	2.233	2.683	1.478
47	ECU_3	167	10	1.000000	2.193	2.232	2.683	1.479
48	ECU_3	204	10	1.000000	2.195	2.233	2.683	1.478
49	ECU_3	047	20	1.000000	2.197	2.236	2.684	1.479
50	ECU_3	165	20	1.000000	2.194	2.233	2.683	1.478
51	ECU_4	332	100	0.999078	2.225	2.238	2.694	1.393
52	ECU_4	333	100	0.999183	2.229	2.242	2.695	1.390
53	ECU_4	439	1,000	0.999614	2.227	2.239	2.695	1.390
54	ECU_4	43A	1,000	0.999614	2.227	2.239	2.694	1.390
55	ECU_4	437	1,000	0.999614	2.227	2.240	2.696	1.391
56	ECU_4	438	1,000	0.999614	2.227	2.241	2.695	1.391
57	ECU_4	091	20	0.999611	2.224	2.237	2.694	1.391
58	ECU_4	23A	100	0.999612	2.223	2.236	2.694	1.391
59	ECU_4	430	100	0.999612	2.227	2.240	2.694	1.393
60	ECU_4	434	100	0.999612	2.227	2.240	2.695	1.392
61	ECU_4	2F1	1,000	0.999612	2.231	2.242	2.686	1.384
62	ECU_4	092	100	0.999612	2.228	2.241	2.694	1.392
63	ECU_4	59E	1,000	0.999612	2.223	2.237	2.693	1.392
64	ECU_4	435	100	0.999613	2.227	2.240	2.695	1.392
65	ECU_4	386	1,000	0.999613	2.224	2.236	2.696	1.389
66	ECU_4	07E	20	0.999613	2.225	2.238	2.694	1.390
67	ECU_4	217	10	0.999927	2.227	2.240	2.694	1.392
68	ECU_4	4B0	20	0.999926	2.226	2.239	2.694	1.392
69	ECU_4	415	20	0.999925	2.224	2.237	2.694	1.392
70	ECU_4	049	20	0.999926	2.227	2.240	2.694	1.393
71	ECU_4	077	20	0.999926	2.224	2.237	2.694	1.392
72	ECU_4	07D	20	0.999926	2.226	2.239	2.694	1.392
73	ECU_4	07F	20	0.999926	2.224	2.237	2.695	1.392
74	ECU_4	214	20	0.999925	2.227	2.240	2.694	1.391
75	ECU_4	216	20	0.999925	2.226	2.240	2.694	1.392
76	ECU_4	213	20	0.999925	2.223	2.236	2.694	1.391
77	ECU_4	076	500	0.999926	2.228	2.240	2.694	1.391
78	ECU_4	416	100	0.999930	2.225	2.238	2.695	1.391
79	ECU_4	083	100	0.999930	2.222	2.237	2.693	1.395
80	ECU_4	326	100	0.999940	2.228	2.240	2.694	1.391
81	ECU_4	3DA	1,000	0.999990	2.227	2.240	2.692	1.391
82	ECU_4	3E0	1,000	0.999993	2.229	2.241	2.696	1.389
83	ECU_4	3C8	1,000	0.999993	2.229	2.242	2.695	1.391
84	ECU_4	04A	100	1.000080	2.225	2.239	2.694	1.392
85	ECU_4	04B	100	1.000080	2.227	2.240	2.694	1.392
86	ECU_4	04C	100	1.000080	2.225	2.238	2.694	1.391
87	ECU_4	082	19	1.006160	2.227	2.240	2.694	1.391

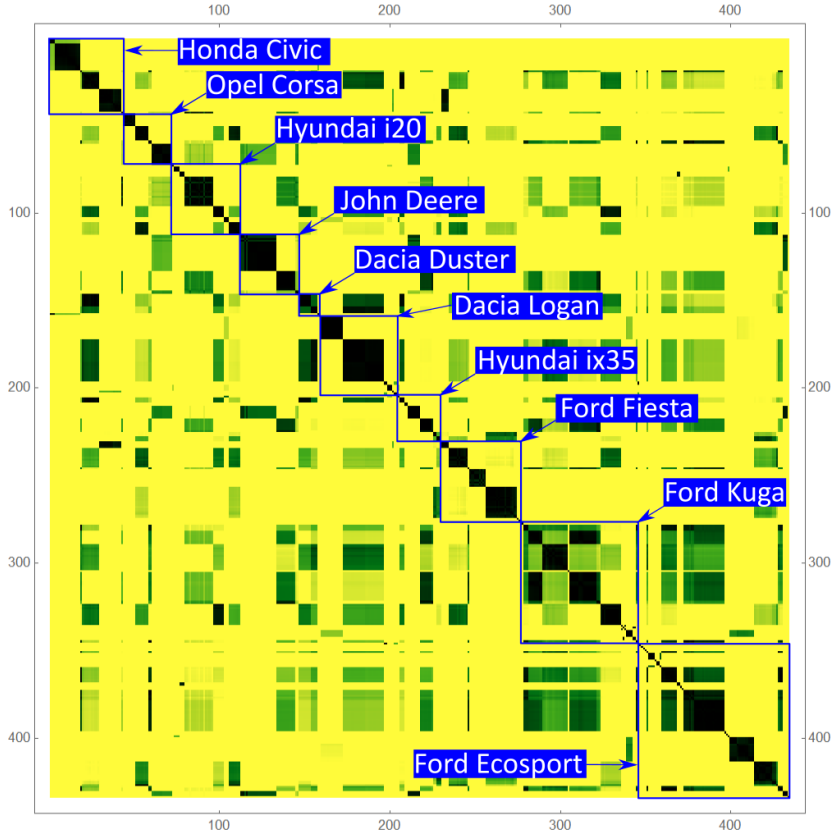
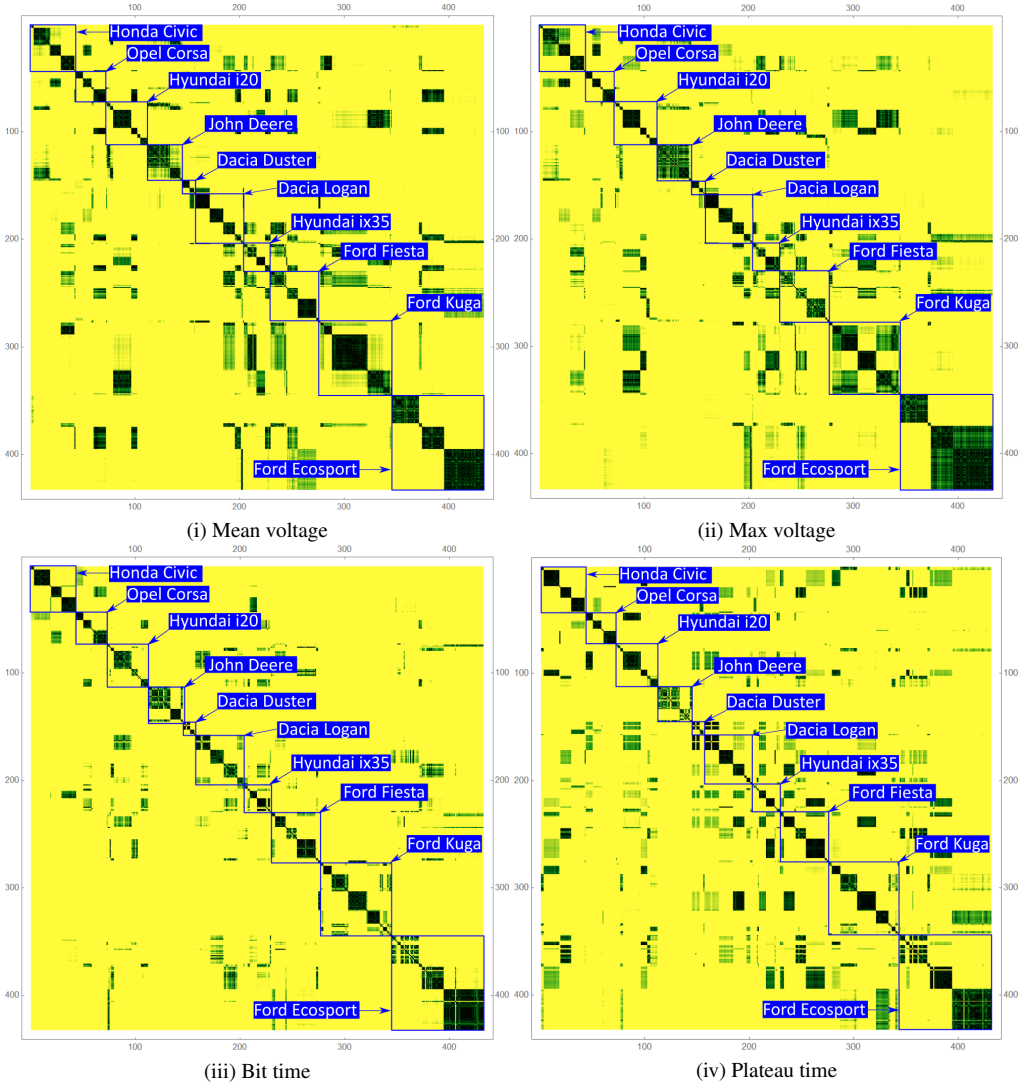


Figure 5.12: Skew heatmaps for the IDs belonging to the ECUs in the vehicles from the experiments

The ECUs from Dacia Logan, which have a difference of only $1ppm$ if clock skew is used and a much higher difference due to voltage characteristics, may be hard to separate using only the clock skew as reference. The same applies to Honda Civic, Dacia Duster, Hyundai i20 and Hyundai ix35 where differences between clock skews for different ECUs are close to $20ppm$. This means that collisions between clock skews are possible due to small intra-distances between ECU_1 and ECU_2 from Honda Civic or ECU_1 and ECU_2 from Dacia Duster.

There is a cleaner separation between most of the ECUs compared to clock skews if mean and maximum voltage are used, as shown in Figures 5.13 (i) and (ii). As shown earlier, in the paragraph that discussed the separation of ECUs inside vehicles, there are ECUs where the values are close to each other, so the intra-distances are small. This applies to ECU_3 and ECU_4 from Honda Civic where the mean voltage and maximum voltage differ with only a few mV. This is also the case mean and maximum voltage



(i) Mean voltage (ii) Max voltage

(iii) Bit time (iv) Plateau time

Figure 5.13: Mean voltage (i), Max voltage (ii), Bit time (iii) and Plateau time (iv) heatmaps for the 433 IDs belonging to the 54 ECUs in the vehicles from the experiments

data is compared between ECUs from different vehicles. Thereby, there is a small inter-distance between ECU_3 and ECU_4 from Honda Civic and ECU_2 from Opel Corsa. The same goes for ECU_4 from Opel Corsa and ECU_3 in Ford Ecosport. Even though the bits extracted from different vehicles have a distinctive pattern with regards to rise time, stabilization time for the voltage, etc., this would require machine learning algorithms to be trained so they could clearly separate the transmitters using voltage patterns. Neverthe-

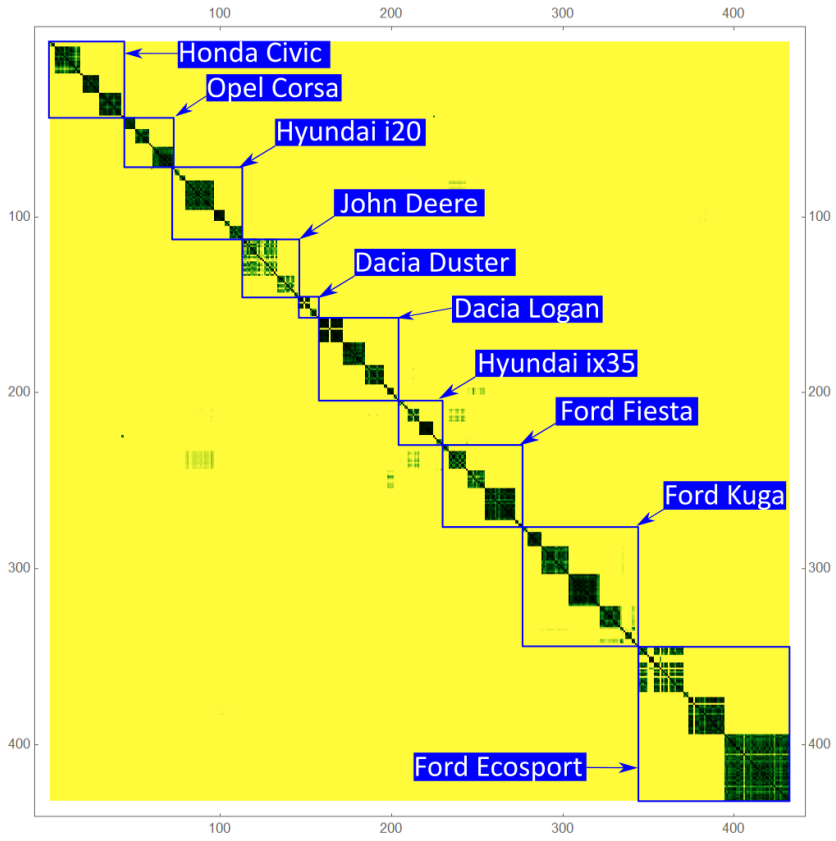


Figure 5.14: Combined mean-max-bit-plateau heatmaps for the IDs belonging to the ECUs in the vehicles from our experiments

less, it was not in scope of this work, since the focus is on the computation of statistical features from the voltage sample dataset.

A cleaner separation between ECUs, as shown in Figures 5.13 (iii) and (iv), can be done using bit and plateau time since they depend on voltage characteristics and also on the clock variations between the transmitters. There are more collisions in the plateau time heatmap (ii) compared to the bit time heatmap (i). This is mostly visible for inter-distances between ECUs that are from different vehicles compared to those from the same vehicle. There are small overlaps which are visible for ECU_4 and ECU_6 from Hyundai ix35 or ECU_1 and ECU_5 from Ford Fiesta. This also applies to ECU_2, ECU_4, ECU_7 and ECU_8 from Ford Kuga or ECU_2 and ECU_3 from Ford Ecosport. There are small inter-distances for the bit time between ECU_2 and ECU_5 from Honda Civic and ECU_1 and ECU_4 from Opel Corsa. Since there are no overlaps between ECUs from Opel Corsa and Dacia Duster, inter-distances are high and intra-distances are low, so the ECUs can

Table 5.12: Differences of physical characteristics after 1 hour of runtime for Honda Civic

No.	ECU	ID	Cycle	$\Delta\mathbb{C}_{skew}$	$\Delta\mathbb{V}_{mean}$	$\Delta\mathbb{V}_{max}$	$\Delta\mathbb{T}_{bit}$	$\Delta\mathbb{T}_{plat}$
1	ECU_1	039	40	0.000007	0.014	0.014	0.011	-0.015
2	ECU_1	305	100	0.000007	0.013	0.012	0.011	-0.011
3	ECU_1	401	300	0.000007	0.016	0.016	0.010	-0.003
4	ECU_2	1A6	20	0.000005	-0.047	-0.050	0.003	-0.003
5	ECU_2	21E	40	0.000005	-0.044	-0.047	0.003	-0.003
6	ECU_2	221	40	0.000005	-0.042	-0.044	0.004	-0.003
7	ECU_2	294	40	0.000005	-0.045	-0.048	0.002	-0.003
8	ECU_2	295	40	0.000005	-0.044	-0.047	0.002	-0.004
9	ECU_2	309	100	0.000005	-0.048	-0.051	0.005	-0.003
10	ECU_2	372	100	0.000005	-0.047	-0.050	0.004	-0.003
11	ECU_2	374	100	0.000005	-0.048	-0.052	0.000	-0.002
12	ECU_2	377	100	0.000005	-0.049	-0.051	0.002	-0.003
13	ECU_2	378	100	0.000005	-0.048	-0.050	0.002	-0.003
14	ECU_2	386	100	0.000005	-0.041	-0.043	0.005	-0.003
15	ECU_2	405	300	0.000005	-0.040	-0.041	0.002	-0.002
16	ECU_2	428	300	0.000005	-0.040	-0.040	0.005	-0.003
17	ECU_2	42D	300	0.000004	-0.040	-0.042	0.004	-0.003
18	ECU_2	42E	300	0.000004	-0.040	-0.041	0.004	-0.004
19	ECU_3	18E	10	0.000010	0.059	0.058	0.010	-0.003
20	ECU_4	091	10	0.000011	0.002	0.004	0.010	0.002
21	ECU_4	19B	10	0.000012	-0.002	-0.001	0.010	0.002
22	ECU_4	1A4	20	0.000012	0.003	0.004	0.010	0.002
23	ECU_4	1AA	20	0.000012	0.004	0.004	0.011	0.002
24	ECU_4	1B0	20	0.000012	0.002	0.003	0.012	0.002
25	ECU_4	1D0	20	0.000012	0.002	0.003	0.011	0.002
26	ECU_4	1EA	20	0.000015	0.002	0.003	0.013	0.002
27	ECU_4	255	40	0.000015	0.003	0.003	0.010	0.001
28	ECU_4	3D9	200	0.000018	0.005	0.006	0.014	0.001
29	ECU_4	406	300	0.000019	0.004	0.002	0.013	-0.003
30	ECU_5	13C	10	0.000023	0.035	0.035	0.010	0.001
31	ECU_5	158	10	0.000023	0.034	0.035	0.010	0.002
32	ECU_5	17C	10	0.000023	0.036	0.037	0.010	0.001
33	ECU_5	1DC	20	0.000023	0.035	0.036	0.008	0.001
34	ECU_5	1ED	20	0.000023	0.038	0.038	0.009	0.001
35	ECU_5	320	100	0.000023	0.040	0.044	0.011	0.001
36	ECU_5	324	100	0.000024	0.039	0.036	0.008	-0.001
37	ECU_5	328	100	0.000024	0.035	0.035	0.006	0.001
38	ECU_5	3D7	200	0.000023	0.039	0.040	0.020	0.001
39	ECU_5	400	300	0.000023	0.039	0.037	0.008	-0.001
40	ECU_5	40C	300	0.000023	0.037	0.036	0.012	0.001
41	ECU_5	454	300	0.000024	0.039	0.038	0.009	0.001
42	ECU_5	465	300	0.000024	0.037	0.038	0.010	0.002
43	ECU_6	156	10	0.000017	0.007	0.005	0.016	-0.013

be clearly separated using the bit time. Similar overlaps that were determined for the bit time apply to the plateau time as it happens for ECU_1 and ECU_3 from Dacia Logan, ECU_1 and ECU_2 from Ford Ecosport or ECU_1 and ECU_5 from Ford Fiesta. There are overlaps for plateau time between ECU_5 from Hyundai i20 and ECU_2 from Honda Civic or ECU_4 from Hyundai i20 and ECU_1 from Opel Corsa due to small inter-distances. The same goes for ECU_2 from Ford Fiesta, ECU_3 from Dacia Logan, ECU_4 from Hyundai ix35 and ECU_5 from Ford Kuga that have a similar plateau time.

In case all four voltage features are mixed, by using mixed physical characteristics, as shown in Figure 5.14, almost all overlaps between ECUs are reduced, with an increase in the inter-distances but also in the intra-distances. Some overlaps which still remain, but with a higher inter-distance than single separation heatmaps, are between Hyundai i20 and Ford Fiesta, Hyundai ix35 and Ford Fiesta or Dacia Logan and Ford Fiesta. There are small overlaps between IDs from Ford Kuga and Ford Ecosport ECUs that can be seen, but their effect is minor. This means that, by using only one or two voltage

Table 5.13: Differences of physical characteristics after 1 hour of runtime for Ford Fiesta

No.	ECU	ID	Cycle	$\Delta\mathbb{C}_{\mathrm{skew}}$	$\Delta\mathbb{V}_{\mathrm{mean}}$	$\Delta\mathbb{V}_{\mathrm{max}}$	$\Delta\mathbb{T}_{\mathrm{bit}}$	$\Delta\mathbb{T}_{\mathrm{plat}}$
1	ECU_1	023	100	-0.000145	0.020	0.023	0.009	-0.004
2	ECU_1	04A	100	-0.000145	0.019	0.021	0.009	0.001
3	ECU_1	04B	100	-0.000145	0.018	0.020	0.007	0.003
4	ECU_1	460	100	-0.000147	0.025	0.029	0.013	-0.001
5	ECU_2	073	10	0.000013	0.011	0.011	0.028	-0.001
6	ECU_2	090	10	0.000013	0.011	0.011	0.024	0.000
7	ECU_2	20E	10	0.000015	0.010	0.010	0.028	-0.001
8	ECU_2	20F	10	0.000015	0.010	0.009	0.027	-0.001
9	ECU_2	211	10	0.000015	0.010	0.010	0.018	0.000
10	ECU_2	212	100	0.000017	0.015	0.015	0.023	0.000
11	ECU_2	213	20	0.000015	0.009	0.008	0.032	0.000
12	ECU_2	215	20	0.000017	0.010	0.010	0.015	0.000
13	ECU_2	216	20	0.000017	0.010	0.010	0.016	0.000
14	ECU_2	2C3	1,000	0.000019	0.009	0.008	0.012	-0.001
15	ECU_2	4B0	10	0.000017	0.010	0.009	0.029	0.001
16	ECU_3	150	25	0.000009	0.003	0.004	0.004	0.000
17	ECU_4	190	20	-0.000119	0.020	0.017	0.017	-0.007
18	ECU_4	275	100	-0.000118	0.017	0.015	0.002	-0.005
19	ECU_4	400	100	-0.000118	0.017	0.014	0.019	-0.005
20	ECU_4	405	100	-0.000119	0.021	0.020	0.017	-0.002
21	ECU_4	430	100	-0.000119	0.020	0.017	0.016	-0.005
22	ECU_4	432	100	-0.000113	0.014	0.010	0.010	-0.006
23	ECU_4	433	100	-0.000108	0.019	0.015	0.018	-0.009
24	ECU_4	4E3	30	-0.000118	0.020	0.018	0.013	-0.005
25	ECU_4	2C1	1,000	-0.000106	0.013	0.014	0.007	-0.002
26	ECU_4	4F2	1,000	-0.000116	0.023	0.022	0.019	-0.007
27	ECU_5	0FD	20	0.000019	0.022	0.020	0.008	-0.002
28	ECU_5	200	10	0.000020	0.021	0.019	0.009	-0.002
29	ECU_5	201	10	0.000020	0.022	0.021	0.008	-0.002
30	ECU_5	203	30	0.000020	0.020	0.018	0.008	-0.002
31	ECU_5	205	10	0.000020	0.023	0.021	0.006	-0.002
32	ECU_5	228	25	0.000019	0.023	0.021	0.000	-0.002
33	ECU_5	231	10	0.000022	0.024	0.022	0.007	-0.002
34	ECU_5	232	10	0.000019	0.022	0.021	0.011	-0.002
35	ECU_5	261	50	0.000022	0.019	0.019	0.002	0.000
36	ECU_5	268	10	0.000022	0.022	0.020	0.007	0.002
37	ECU_5	280	50	0.000022	0.023	0.021	0.004	0.002
38	ECU_5	2BA	100	0.000025	0.022	0.020	0.001	-0.003
39	ECU_5	360	10	0.000022	0.022	0.020	0.007	-0.002
40	ECU_5	364	30	0.000022	0.021	0.020	0.010	-0.002
41	ECU_5	420	100	0.000013	0.021	0.020	0.012	-0.001
42	ECU_5	424	100	0.000017	0.021	0.020	0.003	-0.002
43	ECU_5	428	100	0.000025	0.021	0.020	0.001	-0.003
44	ECU_5	4F1	1,000	0.000028	0.019	0.019	0.017	-0.001
45	ECU_6	080	15	-0.000290	0.016	0.012	-0.002	-0.013
46	ECU_6	240	10	-0.000270	0.015	0.013	-0.003	-0.007

characteristics for fingerprinting purposes, there may be many overlaps, and at least four voltage characteristics are required for a clean separation of ECUs.

5.5.3 Impact of vehicle run-time

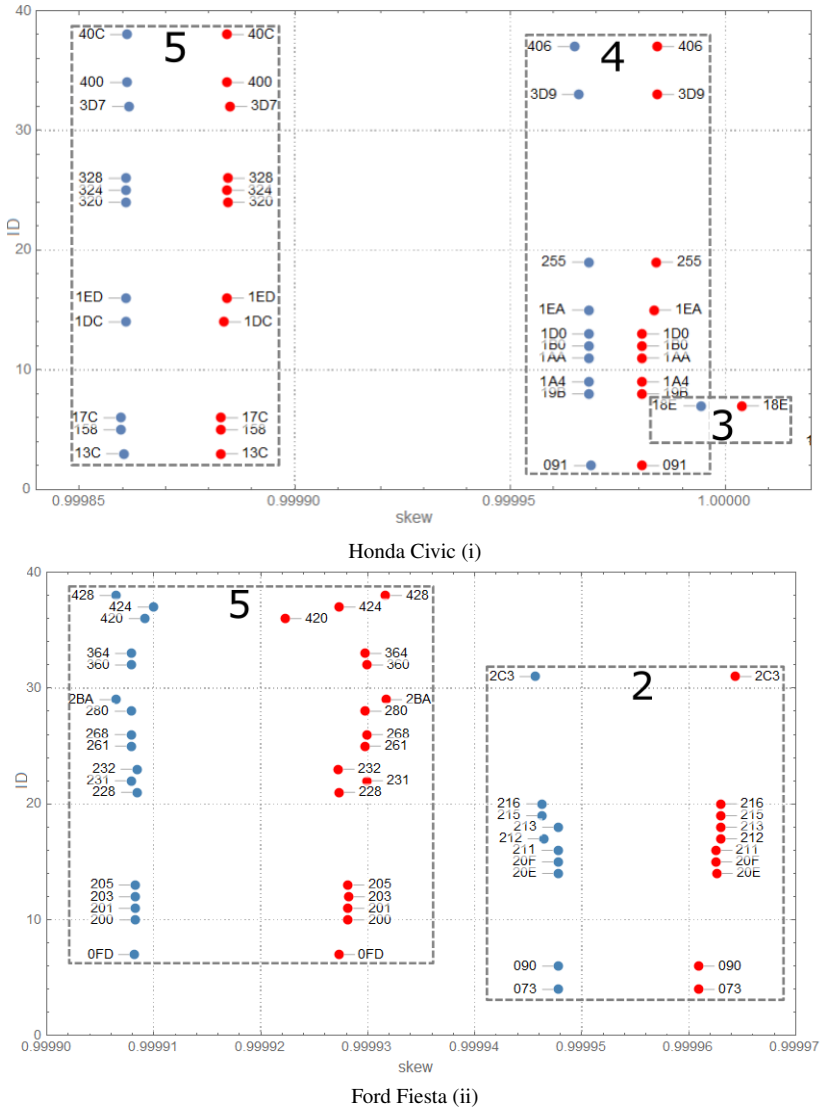
Since the ECU separation is performed using voltage data collected after vehicles were started, what follows is the evaluation of the environmental impact on the clock and voltage characteristics for the Honda Civic and Ford Fiesta passenger vehicles. The reason for this is the consideration that an analysis of environmental change impact on the physical characteristics is deemed necessary. Even though previous works have reported that the physical characteristics have a slight change while the vehicle is running, there are no clear details regarding how they change. The main observation from this analysis is that the variation of physical characteristics does not follow a pattern. The voltage features have either increased or decreased for ECUs from the same vehicle, so no prediction can

be made regarding how they would change over time. The same thing applies to clock skews where values have either increased or decreased based on the samples collected after the 1 hour drive. This means that, in order to cover the changes that impact the physical characteristics over time, extensive studies on the variations of physical characteristics are required for months or even years. Nevertheless, considering one of the use-cases regarding verification of unauthorized ECU presence during annual technical safety inspection, the environmental impact would be minimal if the vehicle is placed in a similar environment, e.g., an authorized service center. The other use-case would be just the generation of an ECU matrix with the transmitters from the in-vehicle buses, based on the access points that are available in a similar fashion to the proposal from [117]. Additional experimental data is presented in Table 5.12 for Honda Civic and Table 5.13 for Ford Fiesta with new columns, Δ, which are emphasized for each physical characteristic. The value inside the Δ columns contains the deviation after 1 hour of driving from the initial value that was measured after vehicle startup.

The clock skew variation for some ECUs in the traces collected after 1 hour of driving the Ford Fiesta and Honda Civic passenger vehicles are shown in Figure 5.15. In the figure, only the clock skew values that have increased after 1 hour are shown. This means that, after this time, either the clock oscillators on these ECUs run a bit faster or the clock oscillator from the CANcaseXL device runs a bit slower. Considering that variations are not uniform for all ECUs, the supposition is that the oscillators from each ECU are mainly responsible for these changes since the reference, i.e., clock of the CANcaseXL, is the same at vehicle startup and after the 1 hour drive.

The mean voltage variation for the ECUs after 1 hour of driving the Ford Fiesta and Honda Civic passenger vehicles are shown in Figure 5.16. The variations for ECUs from Honda Civic are both positive and negative, within 40mV from the initial value, or even at $59mV$ for ECU_3. The ECU from Honda Civic with a negative variation of the mean voltage is ECU_2 while all other ECUs have a positive variation of the same voltage feature. All ECUs from Ford Fiesta have a positive variation of the mean voltage, within $20mV$ from the initial value that tops at $25mV$ for ECU_1. Most of the variations are close to or above the threshold set to $25mV$ for intra-distances and inter-distances.

Following the analyzed results, due to the fact that after 1 hour of driving, since the engine and ECU temperatures increase, the voltage for dominant bits on the CAN bus increases as well while the oscillators have a slight change of their frequency. This is not a general rule since not all mean voltage values increase over time. One of the ECUs from Honda Civic, i.e., ECU_2, had a negative trend of the mean voltage in the samples collected after the 1 hour drive. Since the physical value variation cannot be predicted over time, in order to avoid false reports of intrusions or mis-classification between ECUs, periodic updates of the physical characteristics can help in this regard. Authors from previous research works [27], [127] have proposed re-training of their models with updates of existing fingerprints for intrusion detection systems. It is also possible to update or add new voltage characteristics on intrusion detection systems [27]. This can be done



Honda Civic (i)

Ford Fiesta (ii)

Figure 5.15: Skew variations in the Honda Civic (i) and Ford Fiesta (ii) from a cold start (blue) to 60 minutes driving (red)

using secured updates of the machine learning algorithms through authenticated frames, as shown by authors in [127].

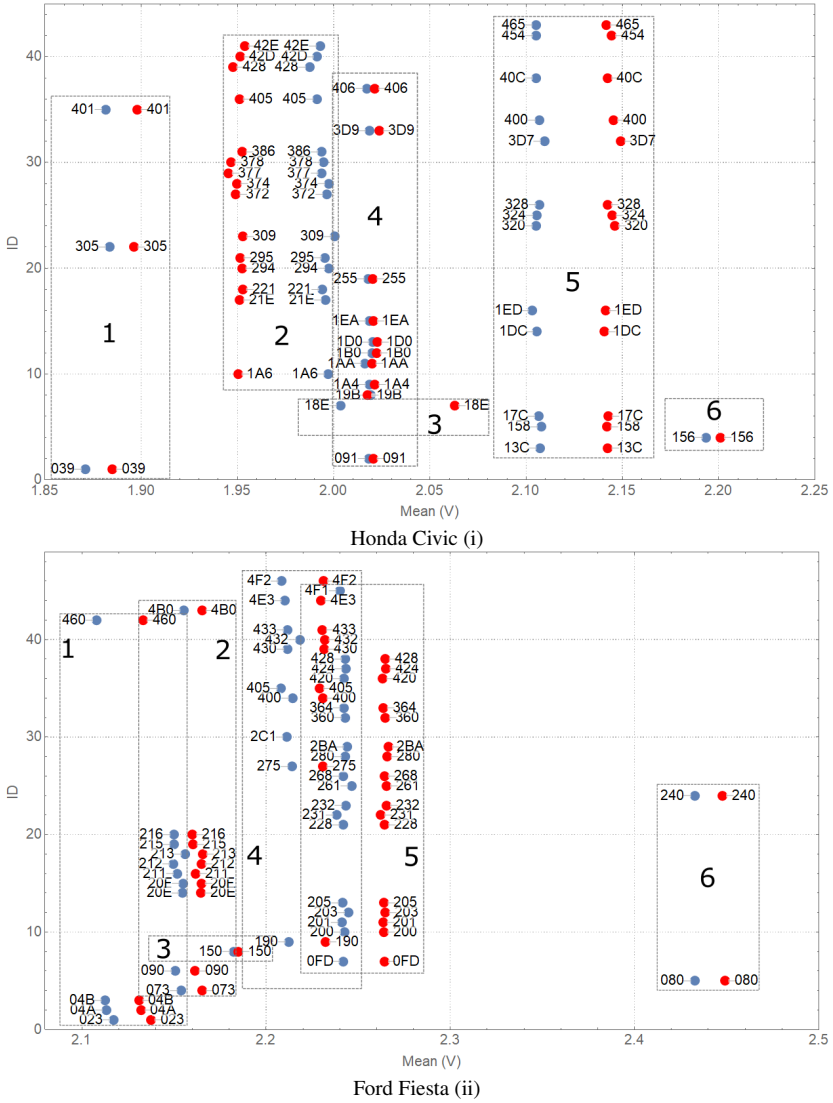


Figure 5.16: Voltage variations in the Honda Civic (i) and Ford Fiesta (ii) from a cold start (blue) to 60 minutes driving (red)

5.6 Concluding remarks

In this chapter, several physical fingerprinting methods were discussed using time and voltage data which was collected from real vehicles. The CAN bus traffic and voltage data samples were collected from 9 passenger vehicles. These were used as inputs for computing the time and voltage features for each frame, based on the frame identifier.

The outcome of the classification was the frame clustering inside ECUs and the clear separation of frames between ECUs. There are 51 ECUs that were identified from the vehicles using the voltage features, while some frames from the same ECU have different clock skews. The reason behind this may be that the ECU is also a gateway for frames transmitted on other vehicle networks. The separation of ECUs based on clock skew and voltage features was also analyzed using inter-distances and intra-distances. There are several overlaps if single voltage features are used for ECU separation, but these are no longer present if multiple voltage features are combined. The impact of vehicle run-time is presented after 1 hour drive for 2 passenger vehicles and the differences discussed. Considering the unpredictable variations, periodic updates of the clock skews and voltage features are necessary, an aspect which was also suggested by other research works.

Chapter 6

An Experimental Setup with Real-world Vehicle Harnesses

This chapter of the thesis is based on two research papers by the author [32], [33]. The first research paper [32] proposes a digital twin for several vehicle level functionalities that are deployed on automotive-grade microcontrollers. Development and integration of MATLAB models on the embedded devices, together with the design of a supporting tool for the CAN interface, are presented in this paper with a comparison of the model outputs with vehicle signals collected from a real car. The second paper [33] uses this setup to perform an analysis of three voltage characteristics for CAN bits. This is needed in order to point out the importance of the wiring harness used on CAN buses regarding voltage fingerprinting techniques.

6.1 Digital Twins for automotive Controller Area Networks

The number of ECUs inside vehicles has increased and each ECU runs specific software components while they require in-vehicle networks to exchange data in real-time. One of the reasons this happened is due to the development of vehicle level functionalities such as adaptive cruise control that require data from multiple systems and sensors to ensure that acceleration and deceleration are safe for the traffic participants. Another change which happened in the automotive industry is related to the upgrade from purely mechanical features such as braking or steering that has been updated by some manufacturers to a combination of mechanical, hardware and software features such as brake-by-wire or steer-by-wire. Following the automotive directions towards autonomous vehicles and autonomous driving there will be many more changes with respect to addition of hardware and software to control the vehicle mechanics or remote software updates [147]. Due to the introduction of more hardware and software components inside the vehicle, the number of vulnerabilities and possible hazards also increased. This allows research groups

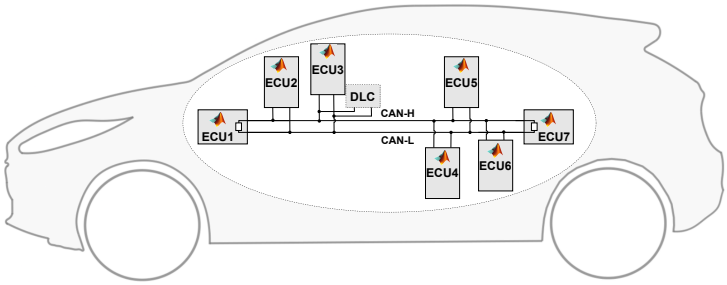


Figure 6.1: Overview of the in-vehicle network CarTwin model

to study already existing and newly developed ECUs that are utilized for several vehicle level functionalities using simulations [148], models [149] and digital twins [150].

Considering that there is an increasing need of digital twin development for the automotive industry, the intended outcome of our work was a car twin which acts as a digital twin for an in-vehicle CAN network. The digital twin includes development boards from Infineon with CAN interfaces that are connected to real CAN network wires from wiring harnesses that were part of a real vehicle. The experimental network is shown in Figure 6.1 with 7 development boards that are placed along the CAN bus, close to the original connectors, and a DLC interface where the CAN interface is connected to the PC. Each development board embeds a MATLAB model which receives inputs and provides real-time outputs using the CAN bus interface. Digital twins of real vehicle CAN networks can be used to examine the security shortcomings in the closest way as possible to a real, practical implementation of the network in the car. Since the number of reported security issues related to the CAN bus is increasing [1, 2, 19], the development of techniques to secure the CAN communication is of utmost necessity. Hence, the design of a vehicle CAN bus digital twin can also pave the road to an improved security analysis of vehicle level functionalities.

6.2 Related works

There are many research papers published in the last decade addressing Controller Area Network topics since they are widely used as communication layers in vehicles. Starting from the initial proposal from Bosch [151] there have been works that studied the bus capacity for message scheduling [152, 153] and also transmission and reception times for frames [154] considering frame arbitration and the error confinement mechanism implemented by CAN nodes. Proposals to enhance the Controller Area Networks have also been published in works like TT-CAN [100] or CANOpen [155]. Others have proposed changes to the original specification in order to increase the communication bandwidth [156, 157]. Newer embodiments of CAN have already been standardized, i.e., CAN-FD

[44, 158], or are in the process of standardization, i.e., CAN-XL [159] for the automation and automotive industries. The updates that the industries bring to the Controller Area Networks show that this bus is still considered as one of the main in-vehicle communication buses. This means that studies related to automotive Controller Area Networks are still of high interest for both the industry and research groups.

The digital twins were first proposed by Michael Grieves in 2002 as part of the product lifecycle management process [160] which was further detailed by the same author in a whitepaper from 2014 [161]. A good definition of a digital twin comes from the International Council on Systems Engineering (INCOSE) which describe it as a high-fidelity model of the system [162]. The authors of [163] mention that there are multiple levels defined for the digital twins, depending on the complexity of their interfaces or the amount of data exchanged with their physical counterparts. Some are preliminary versions of the digital twins called pre-digital twins. The advanced versions are either adaptive digital twins that adapt their interfaces over time or intelligent digital twins with reinforced learning capabilities.

Even though the first digital twins have been practically implemented for manufacturing processes or product lifecycle management activities [164], they are now studied in multiple areas with various use-cases. The idea that a digital twin is an authentic copy of a physical system helps in regards to actively monitoring behaviors or studying particular aspects of the twin, without requiring a connection to the physical system. Some of the applications where the digital twins are developed and researched are the production of cyber-physical systems [165, 166], aerospace equipment [167], oil and gas industry [168], healthcare services [169] or the 5G/6G networks for the Internet of Things (IOT) [169]. In the automotive area, research works propose digital twins for brake systems [170], battery systems [171] or wiring harnesses [172]. In a recently published master thesis [173], its author proposes the realization of a digital twin for the vehicle dynamics of a Toyota Prius vehicle. The design includes single-track, two-track or multi-body MATLAB models to create digital twins for the vehicle dynamics and to capture the model outputs by feeding the vehicle inputs from the CAN bus in real-time. The validation of the digital twin models is done by comparing their outputs with the real-vehicle signal values. There is also increasing interest in studies related to digital twins architectures for autonomous vehicles [174]. As automotive security papers that study different aspects using digital twins, the areas of interest are related to privacy enhancement [175] or the prediction of cybersecurity incidents [176] as well as protection for the infrastructure of intelligent transportation systems [177]. Other related works regarding design details or security overviews for specific automotive systems are briefly described at the end of each paragraph from the following subsection.

6.3 The in-vehicle subsystems from the designed Digital Twin

The CAN bus that serves as basis for the digital twin has a diagnostic interface, so it is the network with the most common access point from the vehicle. There are 7 ECUs are connected to this bus. These are the Accessory Protocol Interface Module (APIM), Power Steering Control Module (PSCM), Instrument Panel Cluster (IPC), Remote Function Actuator (RFA), Restraints Control Module (RCM), Anti-lock Brake System (ABS) and Powertrain Control Module (PCM).

The Accessory Protocol Interface Module (APIM), usually referred to as the SYNC module, is an automotive system with entertainment and multimedia purposes. This system allows vehicle occupants to control the radio channel or the music inputs, connect their mobile devices to the car to allow hands-free voice calls or voice commands. More details regarding this module and its capabilities are provided by the authors of [178].

The Power Steering Control Module (PSCM) controls the steering of the wheels by adapting the position of a column angle using a motor. The input comes from the mechanical rotation of the steering wheel, the measured driver torque and road disturbances. The design of an electric power steering system is presented in [179] with details that include the system model, various block diagrams and equations that define its functionality.

The Instrument Panel Cluster (IPC) provides visual information to the driver regarding the real-time vehicle speed, fuel level, odometer, etc.. This node communicates with other nodes from the vehicle's bus, either directly or through gateways. An in-depth analysis of the instrument cluster is done by authors in [180] with details regarding its functionalities but also attack capabilities that are shown as part of a risk assessment. The work done by authors in [181] is similar, with a summary of attack paths identified for the instrument cluster.

The Remote Function Actuator (RFA) is connected to a remote function receiver and communicates data related to the intelligent key presence. This supports door lock/unlock features but also checks if the intelligent key is in the vehicle or not. There is a multitude of research papers from the past decade that have addresses vulnerabilities and security gaps for keyless door unlock or keyless engine start [182, 183, 184, 185].

The Restraints Control Module (RCM) handles the passive safety of a vehicle with the main functionalities closely related to the passenger occupant seat, seatbelt and airbag. In the event of a car accident, based on acceleration and pressure data, it controls the airbag firing and the pretensioner of the seatbelts. The security of restraint control modules is evaluated in [186] which proposes corrective measures and security verification methods to reduce existing risks and minimize the impact of attacks.

The Anti-lock Brake System (ABS) supports in prevention of tire lock or skidding while driving by maintaining traction between the tires and the road surface. The system is effective for the driver also in case there are conditions on the road that affect the friction between any tire and the road. One of the inputs received from its sensors is

the speed of each wheel. This input is monitored and a hydraulic brake is controlled by the system for each wheel that lost its grip on the road surface due to snow, ice or sand. Authors from [187, 188] analyze the security gaps of anti-lock brake systems and propose counter-measures for them.

The Powertrain Control Module (PCM) handles the engine and transmission for vehicles by controlling fuel injection, real-time emissions and change of the gear for combustion engines. This is a complex module with a multitude of features that were analyzed from a security standpoint by authors in recent works [189, 190], but in the context of electric or hybrid vehicles, where some of the described features may differ from vehicles with petrol or diesel engines.

6.3.1 Wiring schematic and details

Modern cars have multiple wiring harnesses, connectors, wires and their number is growing on an year to year basis as recent studies show [191]. According to the same study, some vehicles include up to 40 distinct harnesses that have over 3,000 wires and more than 700 connectors. The length and weight of the wiring harnesses and connectors would be of 4km and 60kg, which makes them both heavy to integrate in the vehicle and hard to maintain over time.

The experimental setup for CarTwin contains three distinct wiring harnesses from a real-world vehicle with multiple connectors and wires. Since we don't want to bias this research to a particular car manufacturer, as many vehicles have similar architectures, we keep the model anonymous. From all the connectors that were part of the wiring harness, only the connectors that had the CAN bus linked to them are maintained. There is a total of 8 connectors which are part of the setup, 1 is the DLC (OBD-II) diagnostic connector and 7 are for the ECUs that communicated on the CAN bus in the vehicle. An overview of the original network configuration from the real vehicle that we considered with details regarding the ECUs that are connected and the bus termination that is part of the terminal nodes is shown in Figure 6.2. The wire length with respect to the CAN bus that contains the main cables and the stubs is shown in Figure 6.3 for the wiring harnesses from the vehicle. These wiring harnesses are called wiring harness #A, #B and #C. For each of the wiring harnesses, the nodes which are connected to them are shown in the same figure. In case of wiring harness #A, the main wire length is of $140cm$ while the stub length for the PSCM, IPC and DLC is of $15cm$, $50cm$ and $15cm$. The wiring harness #B connects the RCM and RFA to the CAN bus with a main wire length of $195cm$ and the length for the stubs of $120cm$ and $100cm$. The last one, wiring harness #C, provides access on the CAN bus to the ABS and PCM modules with a main wire length of $175cm$ and $25cm$ for the stub that goes to the ABS system. This leads to a total of $510cm$ for the main wire length, without the stubs that are a total of $325cm$ of twisted-pair wires. The terminal nodes, APIM and PCM, also include a 120Ω resistor that connects CAN-H and CAN-L on both ends of the network.

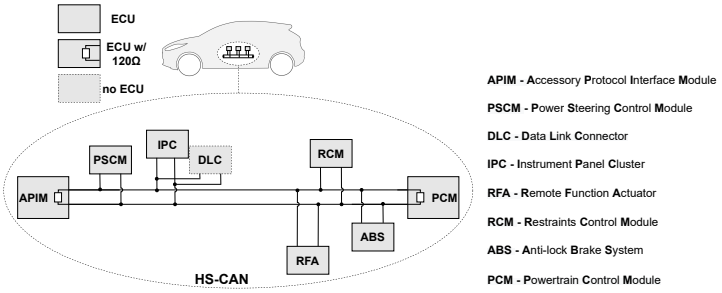
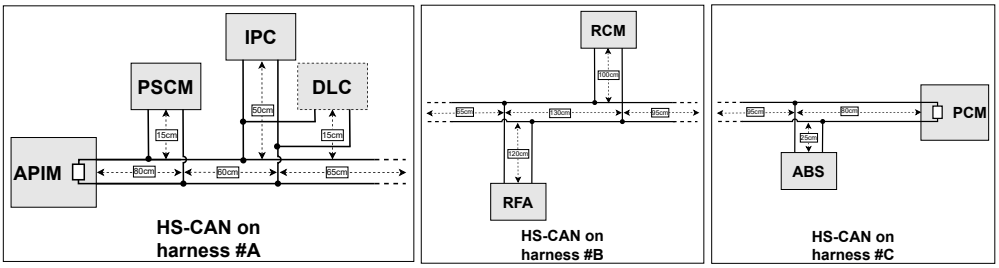


Figure 6.2: Schematic view of the in-vehicle high-speed CAN bus that are used for the digital twin network



(i) CAN bus on harness #A (ii) CAN bus on harness #B (iii) CAN bus on harness #C

Figure 6.3: Wire and stub lengths for the in-vehicle high-speed CAN bus from vehicle harnesses

6.3.2 Design and validation of the models

The overview for the block diagram that contains all the MATLAB models developed in the work together with the input and output signals for each model is shown in Figure 6.4. The models were developed for the ECUs connected to the CAN bus from the wiring harness. These are the APIM, PSCM, IPC, RFA, RCM, ABS and PCM. In addition to the ECU models, the RestBus simulation tool that provides signals required by some models is represented as a block. The RestBus simulation tool [1] provides the CAN signals on the experimental setup, while in MATLAB, the signals provided by this block, are set in the simulation environment. The signals transmitted from this block, marked with orange in the figure, are the driving direction, brake status, buckle status, steering wheel angle and door lock button status. Based on a pre-set target vehicle speed, the vehicle speed marked with green in the figure is computed by the ABS and provided to PSCM and PCM. In what follows next, each model will be described with details regarding the inputs required, input data processing and outputs that are provided.

The inputs used by the Power Steering Control Module (PSCM) model are the steer-

[1] https://www.ni.com/ro-ro/innovations/white-papers/12/the-fundamentals-of-restbus-simulation.html

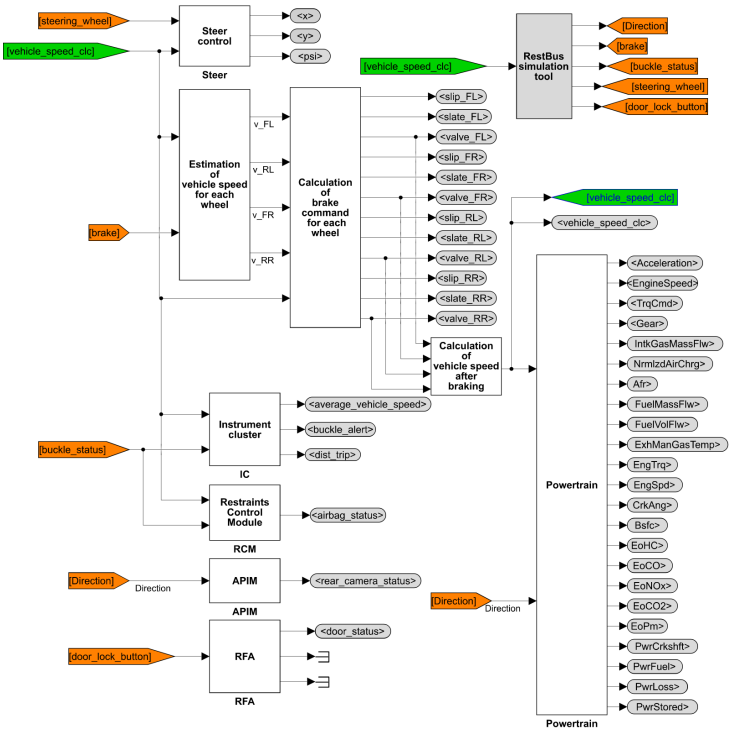


Figure 6.4: Overview of the Simulink model including the seven ECUs

ing wheel angle provided from the RestBus simulation tool and the estimated vehicle speed value provided by the ABS model. Since the Mapped Steering block was available as part of the Simulink library, it was used as the main block for computing the wheel angles. The wheel angles are computed based on interpolation tables that are defined according to the vehicle settings. The output from the PSCM model is represented by the vehicle trajectory defined as $x(t)$, $y(t)$ coordinates and the vehicle rotation angle defined as $\psi(t)$. The values for $x(t)$, $y(t)$ and $\psi(t)$ are computed based on the vehicle speed, $v(t)$ and the wheel angle as shown in Equations 6.1, 6.2 and 6.3. The value for the vehicle speed is divided by 3 in Equation 6.3, since the wheelbase, i.e., distance between the rear and front axles, is set to $3m$. The wheel angle for the right wheel is referenced as $AngR$, while the vehicle speed is referenced as $v(t)$.

$$x(t) = \int \cos\psi \times v(t)\, dt \tag{6.1}$$

$$y(t) = \int \sin\psi \times v(t)\, dt \tag{6.2}$$

$$\psi(t) = \int \tan\left(AngR\right) \times \frac{v(t)}{3}\, dt \tag{6.3}$$

The inputs used by the Anti-lock Brake System (ABS) model are the target vehicle speed value and the brake status from the RestBus simulation tool. The estimated value of the speed for each is computed by the model using the brake status. In case the brake status is active, i.e., the brake is pressed, the wheel speed is decreased based on the output of an integral controller block. The vehicle speed, if the brake is applied, is computed using a proportional controller block and an integral controller block. As inputs to the proportional controller block, either the vehicle speed target is used, when the brake is not applied or 0 when the brake is applied. The value for the vehicle speed is limited to the vehicle speed target if there is no brake applied, or computed using an integral controller block otherwise. The slip for each wheel, $s_x(t)$, is computed based on the vehicle speed and the speed of the wheel $x, x = 1..4$. This is described in Equation 6.4 where $v_x(t)$ is the speed for wheel $x, x = 1..4$, and $v(t)$ is the vehicle speed. The brake state for each wheel is computed based on the determined slip and the vehicle speed value. The brake state values can be set to either $Apply$, $Hold$ or $Release$. Whenever the brake is applied, i.e., brake state is set to $Apply$, the input valve opens while the output valve is closed and the pressure is applied to the wheel. After the valve is open and the applied pressure locks the wheel, in order to keep the wheel locked, the input valve is closed. As a result, the brake is kept pressed on the wheel. In this case, the brake state is set to $Hold$. Based on the slip value for the wheel, the output value is open and the brake is released, allowing the wheel to rotate, i.e., brake state is set to $Release$. Control of the input and output valves is done based on the brake state with the same logic for all four wheels.

$$s_x(t) = \frac{v_x(t) - v(t)}{v(t)} \tag{6.4}$$

The inputs used by the Powertrain Control Module (PCM) model are the vehicle speed value provided by the ABS model and the driving direction from the RestBus simulation tool. The PCM provides multiple outputs that will be explained in what follows. One of the main outputs is the vehicle acceleration, $acc(t)$, computed as the time derivative of the received vehicle speed, $v(t)$, as shown in Equation 6.5. In order to have a stable value of acceleration due to fast transitions of the vehicle speed, two low-pass filters with a filter coefficient set to 0.1 are used in the model before the acceleration is provided outside of the module. Another important output provided by the PCM system is the gear that is computed based on the input vehicle speed. There are 6 gears considered in the model with the vehicle speed threshold to shift between gears of $15km/h$, $30km/h$, $50km/h$, $60km/h$ and $80km/h$. The transition between gears is based on an interpolation table. The engine speed is computed by the PCM model using the shaft vehicle speed, $shaftVS(t)$, axle ratio, $axleRatio$ and gear transmission ratio, $trRatio$, as shown in Equation 6.6. The shaft vehicle speed, $shaftVS(t)$, is determined based on the vehicle speed ($v(t)$) and the wheel radius, which is set to $0.381m$ as shown in Equation 6.7. Using the engine speed and the gear, the PCM provides the engine torque based on a 2-D lookup table. Other signals provided by the PCM from the model are the engine power, air mass flow, fuel flow, exhaust temperature, efficiency and emission performance signals, crankshaft power, fuel input power and power loss. All these signals are provided by the `Mapped SI Engine block` from the Simulink library that contains multiple lookup tables.

$$acc(t) = \frac{dv(t)}{dt} \tag{6.5}$$

$$engineSpeed(t) = shaftVS(t) \times axleRatio \times trRatio \tag{6.6}$$

$$shaftVS(t) = \frac{v(t) \times 25}{3 \times \pi \times 0.381} \tag{6.7}$$

The inputs used by the Instrument Panel Cluster (IPC) model are the vehicle speed value provided by the ABS model and the buckle status from the RestBus simulation tool. The model of the IPC provides three outputs that are the buckle alert (i), trip distance (ii) and average vehicle speed (iii). The trip distance (dist) is computed as integral of the received vehicle speed signal $v(t)$ as shown in Equation 6.8. The average vehicle speed is computed based on the received vehicle speed over time. Since the vehicle speed is sometimes used in m/s in the model due to the implementation of some blocks, it is converted to km/h before the trip distance or average vehicle speed values are computed. The buckle alert is reported as active whenever the vehicle speed is higher than $10m/s$

($36km/h$) and the seatbelt is not buckled. In case of a lower vehicle speed or, if the seatbelt is buckled, the buckle alert becomes inactive.

$$dist(t) = \int v(t)dt \tag{6.8}$$

The inputs used by the Restraints Control Module (RCM) model are the vehicle speed value provided by the ABS model and the buckle status from the RestBus simulation tool. The RCM from the model provides the airbag status as the single output signal. The airbag status is active whenever the vehicle speed is higher than $10m/s$ ($36km/h$) and the seatbelt is buckled. The airbag status is active whenever the seatbelt buckle alert provided by the instrument cluster is inactive and inactive in the other case since the airbag shouldn't work without the seatbelt fastened.

The input used by the Protocol Interface Module (APIM) model is the driving direction from the RestBus simulation tool. The APIM from the model provides a single output signal that is the rear camera status. The rear camera status is active whenever the driving direction is set to reverse and inactive if the driving direction is set to forward. This means that the images captured by the rear camera are shown to the driver whenever the vehicle is driven in a backward direction.

The input used by the Remote Function Actuator (RFA) model is the door lock button from the RestBus simulation tool. The APIM from the model provides a single output signal that is the door status. The door status is updated from unlocked to locked whenever the door lock button is pressed. If the door lock button is continuously pressed, the transition between locked-unlocked or unlocked-locked transition is done every $1s$.

6.3.3 Hardware and software level deployment of the Digital Twin

The signal representation from the models and the RestBus simulation was set according to existing signals from CAN databases that are part of the opendbc GitHub project [2] from commaai. Even though some of the signals, such as the speed or angles, are represented on 15 bits, they are extended to 16 bits so the information can be captured in 2 bytes, but limiting the physical range to 15 bits. This helped us in the implementation of the RestBus simulation tool and the integration and verification of the models on the embedded boards. The RestBus simulation tool was developed in C# and integrated the Vector XL Driver Library [3] to allow transmission, reception and logging of CAN frames in real time using a Vector VN5610 hardware tool. A configurable timer with a microsecond step allows the configuration of the cycle time for the main function that includes the frames transmitted by the RestBus simulation tool. The RestBus simulation tool is configured to transmit specific values for each signal provided to the embedded

[2] https://github.com/commaai/opendbc

[3] https://www.vector.com/int/en/products/products-a-z/libraries-drivers/xl-driver-library/

boards, except for the door lock button, which is always transmitted as pressed in the model. The signals transmitted in the frames follow a format that is interpreted and used by the embedded boards as input for the integrated models. Using the "Start transmission" and "Stop transmission" buttons, communication from the RestBus simulation tool can be started, stopped and restarted based on the end-user input. The signals that are transmitted by the tool are shown on the top part of Table 6.1 where the transmitter is represented by the "CAN Tool". The signals that can be configured using the RestBus simulation tool are the vehicle speed target, vehicle direction, brake status, steering wheel angle and buckle status. The vehicle speed target is the value of the vehicle speed that is reached and maintained by the ABS model in case the brake pedal is not pressed. The vehicle direction is the driving direction for the vehicle that can be in a straight or reverse direction. The brake status is used by the ABS model to control the brakes for each wheel and to compute the vehicle speed value after braking. The steering wheel angle is used by the PSCM model to compute the wheel steering angle, vehicle rotation angle and vehicle trajectory values. The buckle status is used by the IPC and RCM models to provide both the buckle alert and airbag status. The real-time communication with the embedded boards can be visualized either in CAN frame format as "Received frames" or using the interpreted signal values that are shown on the top right side of the tool. The graphical user interface of the RestBus simulation tool with the information that was previously described is shown in Figure 6.5.

The integration of the MATLAB models was done on 7 TC275 lite kit embedded boards from Infineon that have an AURIX TC275 microcontroller and a TLE9251VSJ CAN Transceiver. All boards have a USB interface used for programming the binaries using the Infineon software development environment. They also include a CAN interface with 120Ω resistor between CAN-H and CAN-L. The microcontroller has three individual cores with a core operating frequency of 200MHz. The internal memory of the microcontroller is of 4MB of Flash and 472KB of RAM. According to the datasheet, these microcontrollers are usually used as main controllers for automotive safety systems related to braking, airbag deployment or engine control. A TC275 lite kit is used as an embedded device in the experimental setup as shown in Figure 6.6 (i). The pins used for CAN communication from the embedded boards are P20.8 and P20.7, which are the CAN TX and CAN RX lines. These lines are traced to the TLE9251VSJ CAN Transceiver on the embedded board, so they allow the integration of the microcontroller on the CAN bus using 2.54mm 2-pin male connector. The P20.6 pin state was configured to LOW since this line is connected to the standby input of the TLE9251VSJ CAN Transceiver, which is active HIGH. Since the physical CAN bus requires only two termination resistors, the 120Ω resistors that were connected between CAN-H and CAN-L were removed from 5 boards that integrated the MATLAB models for PSCM, IPC, RCM, RFA and ABS. The 120Ω resistor was maintained on the nodes from the bus ends which are the APIM and PCM. Physical connection of CAN wires from the wiring harness to the embedded boards was possible after the wires were cut from the original vehicle connectors and

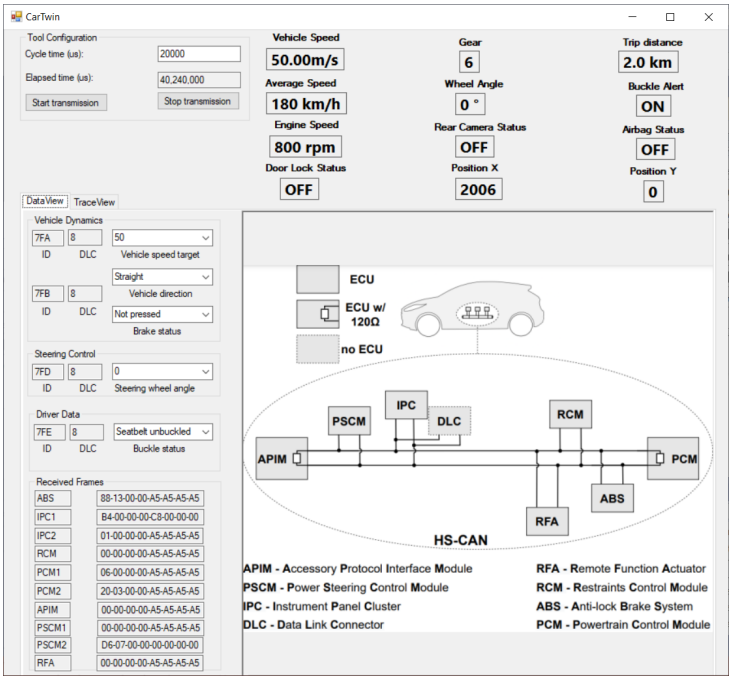
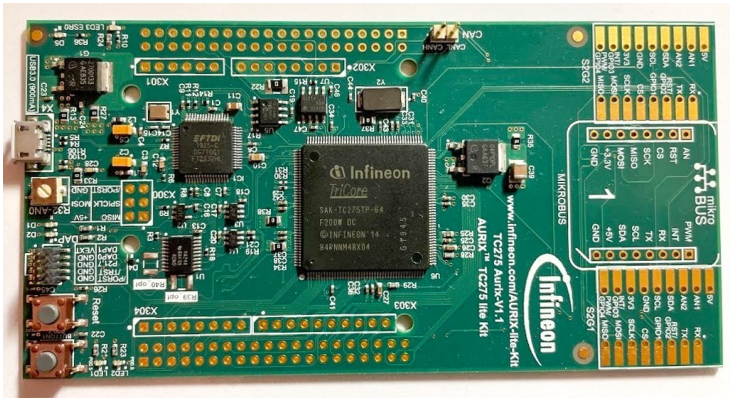


Figure 6.5: User interface for the transmission and visualization of CAN signals

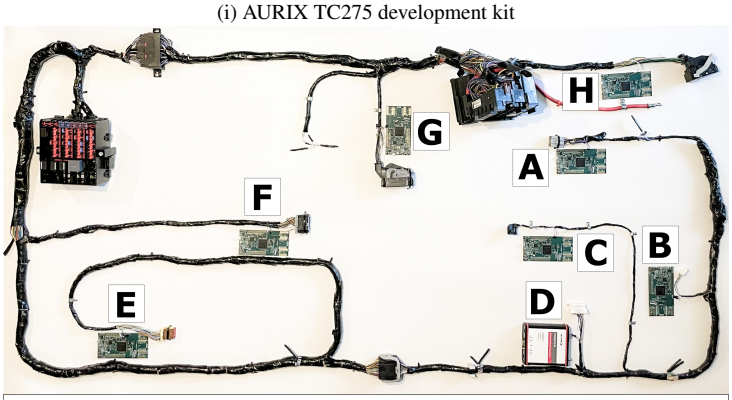
had 2.54mm female headers soldered in order to mate the existing connectors from the embedded boards. The visual representation of the experimental setup is shown in Figure 6.6 (ii) with labels for each embedded device and the MATLAB model it integrates and the VN5610 hardware tool that is connected close to the DLC port.

The information regarding the integration of CarTwin models in the embedded boards is described in what follows. The models from MATLAB/Simulink can be generated as C or C++ code using the Simulink Embedded Coder feature from MATLAB [4]. This functionality from MATLAB performs source code generation from existing models, but specific settings are required in the model if the code is intended for embedded devices. The first setting is related to the solver type from the MATLAB environment that had to be set to "discrete states", to allow setting of the cycle time. The next setting is related to the cycle time of the task executed on the embedded device is the step time in the model that was set to $20ms$. In order to execute the generated code outside of the MATLAB environment without any dependencies, all continuous blocks used by Simulink need to be changed to discrete. Since the model uses discrete blocks, the configured step time is used both in the model simulation and generated source code by the discrete blocks. The final setting is related to the embedded board target for the model that was set to "Infineon" and "Tricore" because AURIX TC275 microcontrollers are used. This allowed

[4] https://www.mathworks.com/help/ecoder/



(i) AURIX TC275 development kit



(ii) Experimental setup of the vehicle bus network digital twin

Figure 6.6: Aurix node and experimental setup for CarTwin

the generated source code to be aligned with the embedded board compiler with respect to the endianness and utilized data types. An additional change related to updates in all MATLAB/Simulink models of continuous time blocks to discrete time blocks is also required. Without this change, the generated source code relies on specific dependencies that are available only where the MATLAB tool is installed. The source code generated from the updated model that has all the continuous time blocks replaced by discrete time blocks is portable to the Infineon embedded device.

The generated source code was integrated in the template projects available as part of the build system from Infineon, i.e., AURIX studio. The project configuration was set for the AURIX TC275 microcontroller since the target devices are the TC275 lite kit embedded boards. The template project includes all required drivers that allow configuration of the peripherals, such as the CAN controller or other functionalities like internal timers. The generated MATLAB/Simulink model contains two functions, one that initializes all

the structures with default data and a runnable that updates the structures based on received input data and provides the output. The initialization function was placed after the initialization part of the microcontroller, where both the CAN controller and internal timer were also initialized. The CAN controller was configured to communicate using a 500Kbit/s bit rate using the P20.8 and P20.7 pins as CAN TX and CAN RX interfaces. The state of P20.6 pin was configured to output, LOW, to enable the CAN transceiver to switch from standby to normal mode. Frame receival is configured to happen during interrupts that are set whenever a frame is received from the CAN bus. The internal timer of the microcontroller was configured to trigger an interrupt every $20ms$, which is in line with the step time configured in MATLAB for the model. The interrupt service routine (ISR) updates a flag that is used in the main function to execute the model runnable and to send the latest output on the CAN bus. The information is sent using one or multiple CAN frames depending on the signals produced by the model. The signals computed by the ABS model are the slip value and valve status for all four wheels, while the input required by the model is the brake status signal sent by the RestBus simulation tool. The signals computed by the PCM model are the engine speed and gear position while the input required by the model are the vehicle speed provided by the ABS module and the driving direction sent by the RestBus simulation tool. The signals computed by the PSCM model are the vehicle steering offset and trajectory information, while the inputs required by the model are the steering wheel angle sent by the RestBus simulation tool and the vehicle speed provided by the ABS module. The signal computed by the RCM model is the airbag status while the input required by the model is the buckle status signal sent by the RestBus simulation tool. The signals computed by the IPC model are the average vehicle speed, trip distance and buckle alert, while the inputs required by the model are the vehicle speed provided by the ABS module and the buckle status signal sent by the RestBus simulation tool. The signal computed by the RFA model is the door lock status, updated every 1 second, based on the door lock button input. For the RFA model, the door lock button input is hardcoded as always pressed in the software implementation. The signal computed by the APIM model is the rear-view camera status, while the input required by the model is the driving direction sent by the RestBus simulation tool. All input and output signals from the model and the RestBus simulation tool with their maximum data size are summarized in Table 6.1. As already mentioned, the first part of the table which has the "Transmitter" set to CAN tool contains CAN signals and frame identifiers that contain the signals transmitted by the RestBus simulation tool. The other signals and frames from the table that have the "Transmitter" set to PCM, ABS, PSCM, RCM, IPC, RFA or APIM are transmitted by the nodes that integrate the MATLAB/Simulink models respectively.

Table 6.1: Summary of signals transmitted by the CAN tool and CarTwin nodes

CAN Signal	CAN ID	Transmitter	Data size (bits)
Vehicle speed target	0x7FA	CAN tool	16
Vehicle direction	0x7FB	CAN tool	2
Brake status	0x7FB	CAN tool	1
Steering wheel angle	0x7FD	CAN tool	16
Buckle status	0x7FE	CAN tool	1
Engine speed	0x11	PCM	32
Gear	0x13	PCM	4
Vehicle speed	0x24	ABS	32
Vehicle steering offset	0x30	PSCM	32
Vehicle position X	0x31	PSCM	32
Vehicle position Y	0x31	PSCM	32
Airbag status	0x23	RCM	1
Vehicle average speed	0x21	IPC	32
Trip distance	0x21	IPC	32
Buckle alert	0x22	IPC	1
Door lock status	0x40	RFA	1
Rear camera video status	0x12	APIM	1

6.3.4 Experimental results

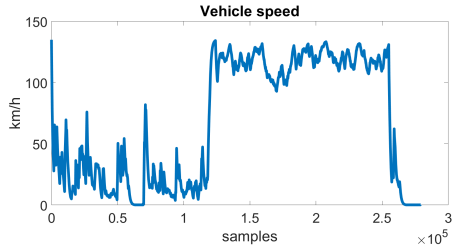
The results related to the integration of the models on the embedded devices and a comparison between the model outputs and real-world vehicle data are presented in what follows. After this, information related to practical use-cases of the models and experimental setup as well as possible updates of the CarTwin are also discussed. Furthermore, a comparison between CarTwin and similar implementations from the automotive areas that use models or digital twins is shown.

After the models were deployed on the embedded boards as part of the software modules executed during runtime, the model integration test was performed. The model integration check consists of verification between outputs from the MATLAB/Simulink models and the embedded boards after providing the same input values. The first step of the integration test was an offline verification on the embedded boards where the input signals used in MATLAB/Simulink were stored as an input array. The values from this array were provided each time the model runnable was executed. The output arrays from the embedded board were stored as a separate array and verified against the output arrays from the MATLAB/Simulink environment. After confirmation that the values from both output arrays are the same, the second step of the integration test was done by providing the same input array was added in a modified version of the C# tool so the input signals were consumed by the boards from the CAN network. The CAN bus communication was logged using the C# tool, so both input values to each embedded board and embedded board output values were recorded. The integration test was considered successful since the output values from MATLAB/Simulink models and the embedded boards were the same. This denotes that the MATLAB/Simulink models are a digital twin of the embedded boards that communicate on the CAN bus.

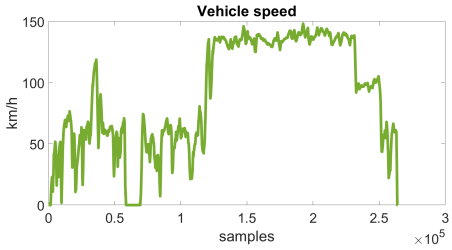
In order to compare the CarTwin with real-world behavior of automotive systems, values for three signals were extracted from a CAN trace recorded for this work, in a passenger vehicle, during a $\sim$40 minute drive. The signals that were extracted are also computed by the ABS, PCM and IPC models in CarTwin. The relevant signals in this

respect are the vehicle speed after braking provided by the ABS model, the engine speed provided by the PCM model and the trip distance provided by the IPC model. Since the brake signal could not be identified in the recorded CAN trace, it was estimated in MATLAB/Simulink based on the vehicle speed variation. In case the vehicle speed was constant or increasing, it was considered that the vehicle accelerates so the brake is not active. When the vehicle speed was decreasing, it was considered that the vehicle decelerates, so the brake is active. In addition to the brake status, the vehicle direction was always sent as straight because there were no reverse driving parts recorded in the real vehicle CAN trace. Since the driving scenario contains an area where the vehicle is driven on the highway, the vehicle speed target input for the model was set to $140km/h$. In this case, this is also the initial speed of the model, before the brake input is taken into account, in contrast with the real vehicle where the initial speed is of $0km/h$. For the CarTwin model, in case the brake status is constantly active, the vehicle speed after braking will decrease down to 0km/h. On the other hand, if the brake status is constantly inactive, the vehicle speed after braking will increase up to or remain at $140km/h$, similar to the cruise control feature from real-world vehicles. The steering wheel angle was always sent as 0, since it could not be extracted from the real vehicle CAN trace while the buckle status was sent as buckled since the driver had fastened the seatbelt before driving in the real-world drive. An overview of the vehicle speed, engine speed and trip distance outputs from the CarTwin models is shown in Figure 6.7 (i), (iii) and (v), while the vehicle speed, engine speed and trip distance outputs from the real-world vehicle are shown in Figure 6.7 (ii), (iv) and (vi).

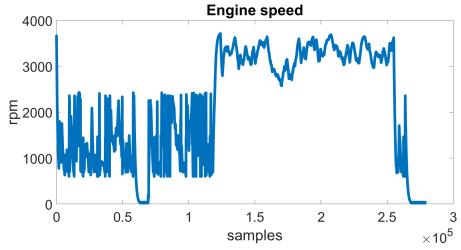
The vehicle speed and engine speed are transmitted in the same CAN frame in the real-world vehicle with a cycle time of $10ms$. The odometer is transmitted in a different CAN frame in the real-world vehicle with a cycle time of 1s. These were the CAN frames used to extract the relevant signals from the traces recorded from the vehicle. The interpretation of data bytes from the frame, as recorded by the Vector XL Driver Library, was possible by using a python script that was developed for this work. The mapping of data bytes to signal values in km/h, rpm or km is possible using reverse engineering methods that are available online. The information regarding the vehicle model and the signal to frame mapping is not disclosed since it is considered property of the automaker. In order to align the odometer value from the real-world vehicle to the trip distance from the CarTwin setup, the trip distance from the vehicle is represented as the difference between the actual odometer value and the initial odometer value from the trace. During the $\sim$40 minute drive there were more than $50{,}000$ samples used for the vehicle speed and engine speed signals and more than $2{,}500$ samples used for the odometer value / trip distance signal. This is visually depicted in Figures 6.7 (ii), (iv) and (vi). The number of samples from the CarTwin models is aligned to the one from the vehicle trace except for the trip distance where the number of samples is the same for vehicle speed, engine speed and trip distance. The vehicle trace contains two driving conditions, one with lower speeds that vary between $0km/h$ and $60km/h$ with a small



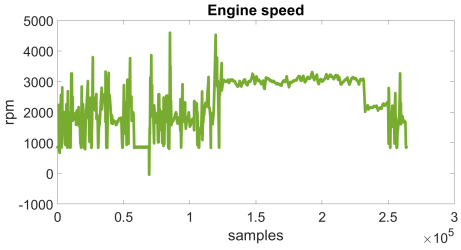
(i) Vehicle speed computed by the models



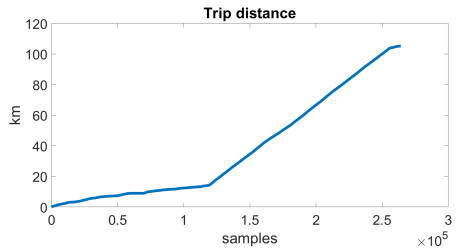
(ii) Vehicle speed collected from the car



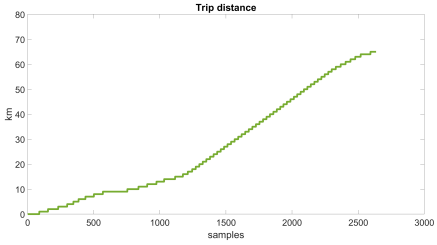
(iii) Engine speed computed by the models



(iv) Engine speed collected from the car



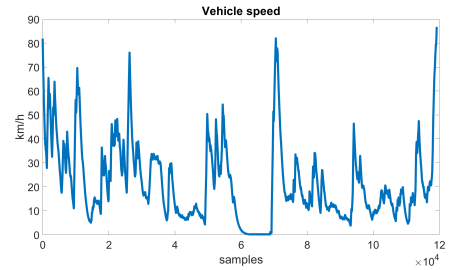
(v) Trip distance computed by the models



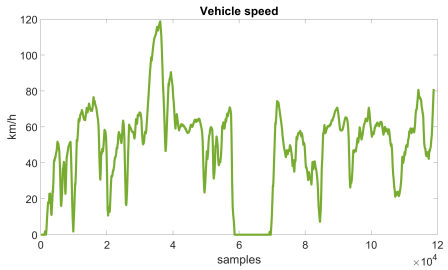
(vi) Trip distance based on odometer value collected from the car

Figure 6.7: Signals computed by CarTwin models (left) and signals collected from a car (right)

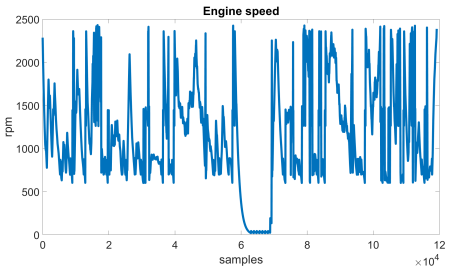
peak over $100km/h$ for the first $\sim$100, 000 samples and the other with an approximately constant value of around $130km/h$ for the next 150,000 samples. The engine speed value from the first part varies between $1{,}000rpm$ and $4{,}500rpm$ while the engine speed from the second part is close to $3{,}000rpm$ for around 100,000 samples until it drops to $2,000rpm$ and down to $1{,}000rpm$ at the end of the trace. Due to a lower vehicle speed in the first part, the trip distance computed based on the odometer value has a low increase rate. Once the vehicle speed reaches $130km/h$ and is maintained for the next $\sim$100,000 samples, the trip distance computed based on the odometer value has a higher increase rate. In order to provide a detailed comparison between the CarTwin model outputs and the real-world vehicle signals, the two parts are separated and explained in the following paragraphs. The visual depiction of the relevant signals from the CarTwin model and real-world vehicle trace from Figure 6.7 is split in Figure 6.8 and Figure 6.9 based on the driving location and driving conditions. The vehicle driving location was on local roads in the first part of the vehicle trace, as shown in Figure 6.8 and on the highway on the



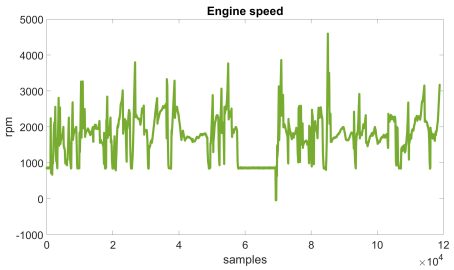
(i) Vehicle speed computed by the models



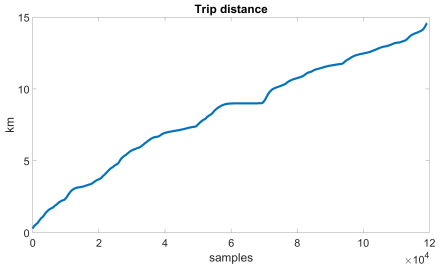
(ii) Vehicle speed collected from the car



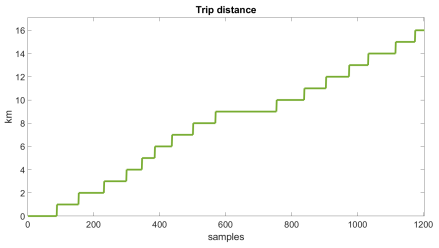
(iii) Engine speed computed by the models



(iv) Engine speed collected from the car



(v) Trip distance computed by the models

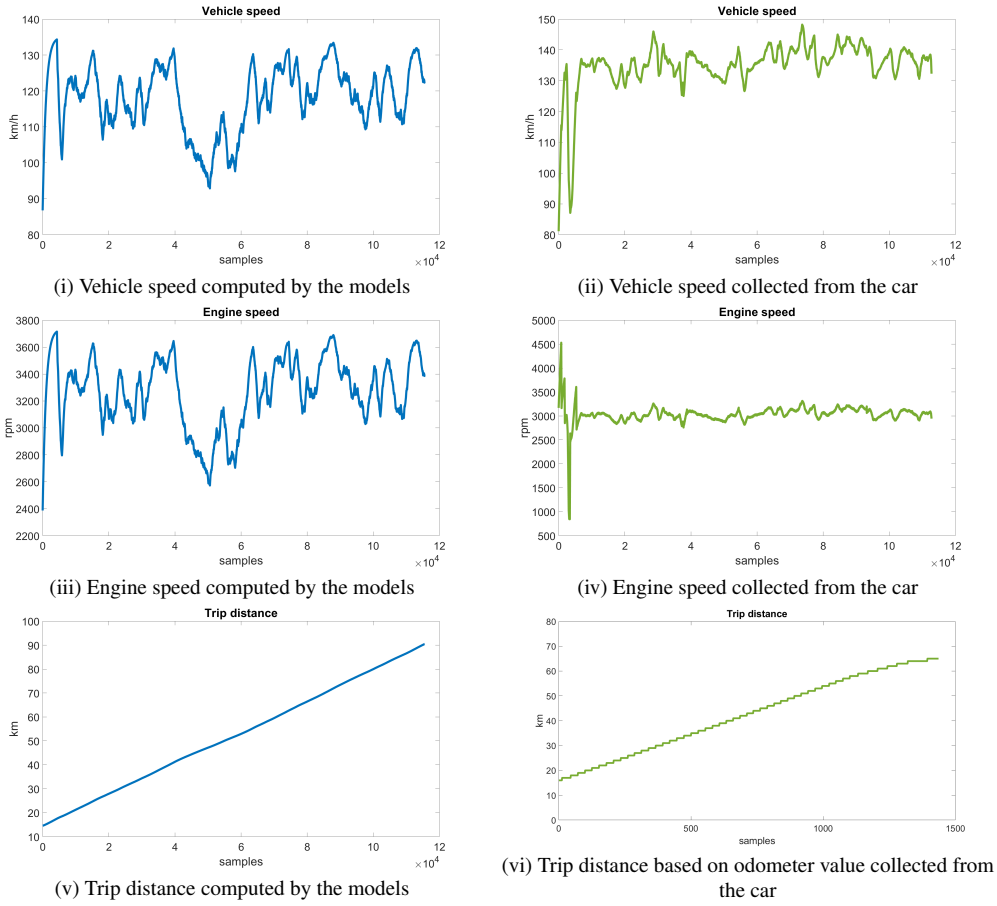


(vi) Trip distance based on odometer value collected from the car

Figure 6.8: Signals computed by CarTwin models (left) and signals collected from a car (right) on local roads

second part of the vehicle trace, as shown in Figure 6.9.

Local roads. The vehicle signals collected while the vehicle driving location was on local roads, which means in the city and the rural surroundings, before entering on the highway, are shown on the right side in Figure 6.8. The output signals from the CarTwin model using the inputs described in the previous paragraph are shown on the left side of Figure 6.8. The vehicle speed variation interval is $[0km/h, 85km/h]$ for the CarTwin output, as shown in Figure 6.8 (i), and $[0km/h, 80km/h]$ for the real vehicle output, except for an increase to $120km/h$ that happened during a car overtake as shown in Figure 6.8 (ii). At some point the vehicle was stationary and the vehicle speed was $0km/h$ during that time for both the CarTwin model and the passenger vehicle. The engine speed variation interval is $[800rpm, 2500rpm]$ for the CarTwin output, as visually depicted in Figure 6.8 (iii), and of $[900rpm, 2500rpm]$ for the real vehicle output except



(i) Vehicle speed computed by the models

(ii) Vehicle speed collected from the car

(iii) Engine speed computed by the models

(iv) Engine speed collected from the car

(v) Trip distance computed by the models

(vi) Trip distance based on odometer value collected from the car

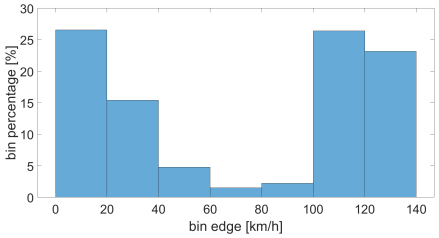
Figure 6.9: Signals computed by CarTwin models (left) and signals collected from a car (right) on highway

for fast increases of the vehicle speed when the engine speed had small peaks at around 4000–$4500rpm$ as shown in Figure 6.8 (iv). Even though the sample rate for the trip distance in CarTwin is different than the one for the odometer from the real vehicle, since the signal is transmitted with a cycle time of $20ms$ in the first case and of $1s$ in the latter, the trend for the signals shown in Figures 6.8 (v) and (vi) is similar and related to variations of the vehicle speed.

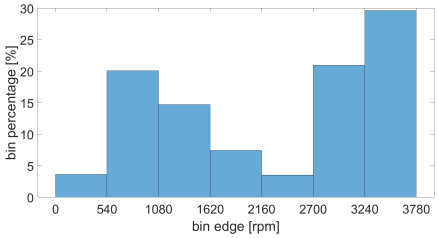
Highway. The vehicle signals collected while the vehicle driving location was on the highway are shown on the right side in Figure 6.9. The output signals from the CarTwin model using the inputs described in the previous paragraph are shown on the left side of Figure 6.9. The vehicle speed variation interval is $[90km/h, 135km/h]$ for the CarTwin output as shown in Figure 6.9 (i) and $[125km/h, 148km/h]$ for the real vehicle output, except for a drop at around $90km/h$ while entering the highway at the beginning of

Figure 6.9 (ii). The engine speed variation interval is [2,600rpm, 3,700rpm] for the CarTwin output as presented in Figure 6.9 (iii) and [2,600rpm, 3,400rpm] for the real vehicle output, except for the engine speed at the beginning of Figure 6.9 (iv) where there engine speed varies between 1,000rpm and 4,500rpm. Even though the sample rate for the trip distance in CarTwin is different than the one for the odometer from the real vehicle, the trend for the signals is very similar and related to variations of the vehicle speed, as presented in Figure 6.9 (v) and (vi).

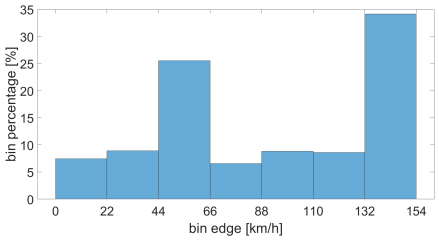
Statistical comparison. In order to measure the accuracy of the CarTwin model, an extension of the initial comparison of its output values and the output values from the real-world vehicle is done by computing the mean values for the sample-by-sample differences between the same signals. Then, the correlation coefficient based on the differences is computed with the support of the MATLAB/Simulink environment to show the correspondence between signals. As a first step in this direction, the histogram distribution of the vehicle speed and engine speed values, together with the differences, are shown in Figure 6.10. Each plot contains 7 bins with specific range intervals that are detailed as follows. For the vehicle speed signal from the CarTwin model, the ranges are defined as [0km/h, 20km/h), [20km/h, 40km/h), [40km/h, 60km/h), [60km/h, 80km/h), [80km/h, 100km/h), [100km/h, 120km/h), [120km/h, 140km/h). For the CarTwin vehicle speed output, more than 20% of the values are either in the first two bins or in the last two bins , as depicted in Figure 6.10 (i). For the vehicle speed signal from the vehicle, the ranges are defined as [0km/h, 22km/h), [22km/h, 44km/h), [44km/h, 66km/h), [66km/h, 88km/h), [88km/h, 110km/h), [110km/h, 132km/h), [132km/h, 154 km/h). For the vehicle speed output from the real-world vehicle, more than 20% of the values are either in the third bin or in the last bin, as depicted in Figure 6.10 (ii). For the engine speed signal from the CarTwin model, the ranges are defined as [0rpm, 540rpm), [540rpm, 1,080rpm), [1,080rpm, 1,620rpm), [1,620rpm, 2,160rpm), [2,160 rpm, 2700 rpm), [2,700rpm, 3,240rpm), [3,240rpm, 3,780rpm). For the CarTwin engine speed output, more than 20% of the values are either in the third bin or in the last two bins, as shown in Figure 6.10 (iii). For the engine speed signal from the real-world vehicle, the ranges are defined as [0rpm, 700rpm), [700rpm, 1,400rpm), [1,400rpm, 2,100rpm), [2,100rpm, 2,800rpm), [2,800rpm, 3,500rpm), [3,500rpm, 4,200rpm), [4,200rpm, 4,900rpm). For the CarTwin engine speed output, more than 40% of the values are in the fifth bin, while more than 15% of the values are in the third and fourth bins, as presented in Figure 6.10 (iv). For the differences, the distributions are shown in Figures 6.10 (v) and (vi). For the vehicle speed values, more than 75% of the values are within the [0km/h, 40km/h) interval, while for the engine speed, more than 80% of the values are within the [0rpm, 1,120rpm) interval. The values for the bin percentages for the vehicle speed and engine speed output signals and the differences between them, together with the bin width for each histogram, are summarized in Table 6.2.



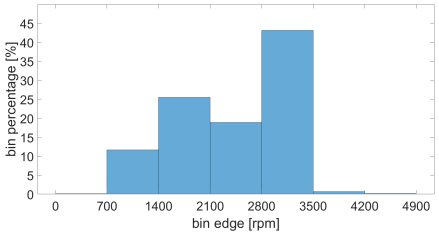
(i) Distribution of vehicle speed values from model



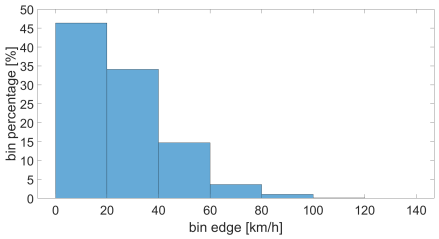
(ii) Distribution of engine speed values from model



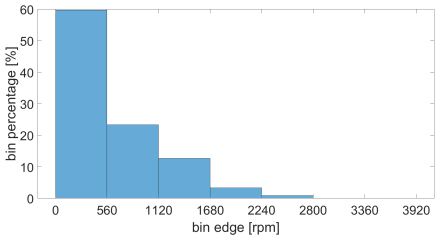
(iii) Distribution of vehicle speed values from vehicle trace



(iv) Distribution of engine speed values from vehicle trace



(v) Distribution of vehicle speed differences



(vi) Distribution of engine speed differences

Figure 6.10: Distribution of values from model output, vehicle trace and differences between them for vehicle speed (left) and engine speed (right)

Table 6.2: Statistical data for distribution of model output, real vehicle signal and their difference

Signal	Bin Width	Bin Percentages [bins 1 to 7]
Vehicle speed (model)	20 [km/h]	27, 15, 5, 2, 2, 26, 23 [%]
Engine speed (model)	540 [rpm]	4, 20, 15, 7, 3, 21, 30 [%]
Vehicle speed (trace)	22 [km/h]	7, 9, 25, 7, 9, 9, 34 [%]
Engine speed (trace)	700 [rpm]	0, 12, 25, 19, 43, 1, 0 [%]
Vehicle speed (difference)	20 [km/h]	46, 34, 15, 4, 1, 0, 0 [%]
Engine speed (difference)	560 [rpm]	60, 23, 13, 3, 1, 0, 0 [%]

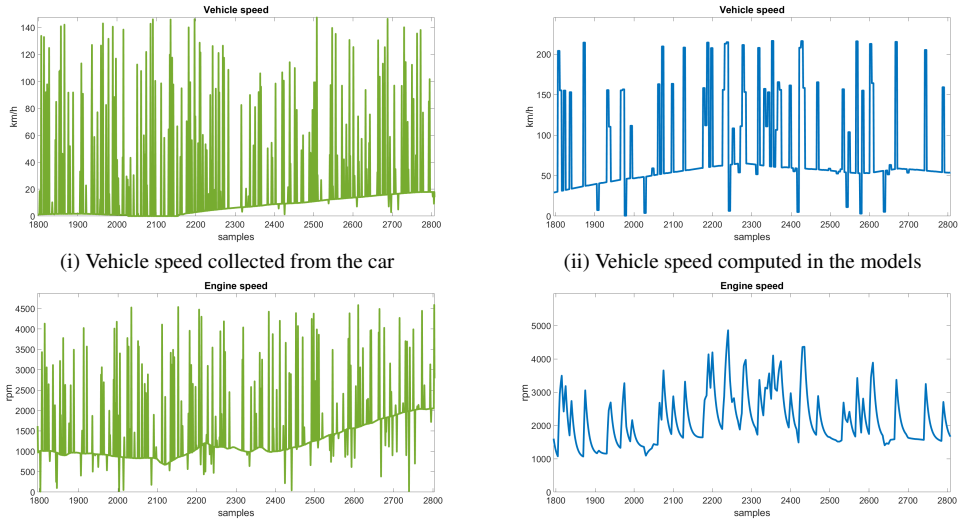
Table 6.3 contains both the mean difference and the correlation coefficient between signals computed using all their samples, in both driving locations, which are the local roads and the highway. Since the CarTwin is a model of vehicle level functionalities that produces synthetic outputs while the signal outputs from the car are produced by real automotive systems, differences between them is expected. The resulted mean difference between the vehicle speed signals is of $25km/h$, while the mean difference for the engine speed signals is of $610rpm$. Since the CarTwin model for the ABS module and the real

system have only one thing in common which is that they receive the same brake signal as input, these differences are somehow expected. The correlation coefficient is of 0.85 between CarTwin vehicle speed and the vehicle speed collected from the car, while the correlation coefficient is of 0.71 between CarTwin engine speed and the engine speed collected from the car. This confirms that there is a good correlation between the realized CarTwin model and the real-world vehicle functionalities related to vehicle speed and engine speed.

Table 6.3: Statistical comparison of the synthetic model outputs with the real vehicle signals

Signal	Range	Mean Difference	Correlation coefficient
Vehicle speed	0–148 [km/h]	25.08	0.85
Engine speed	0–4,597 [rpm]	610.01	0.71

Possible applications. As a primary use-case, the CarTwin model and experimental setup can be used for evaluating cyberattacks on CAN buses. This is a topic that has been studied in the recent years with changes applied to traces generated with local tools or collected from real-world vehicles [192], [193], [194], to include offline attacks with the main purpose of detecting intrusions. The fuzzing attacks that, in the context of CAN, transmit randomized data bytes inside legitimate frames or randomized frames, were also studied in the previous years [195], [196], [197]. In case the attacks are performed directly on CAN buses from the vehicle, which was also studied in the past [45], the risk of damaging the car or affecting the driver, passenger and traffic safety is highly increased compared to attacks on experimental setups that produce no damage to the vehicle and no harm to human beings. This means that performing offline attacks is a good way to study cyberattacks for the CAN bus, but there is one key aspect that affects the output of offline attacks compared to real-time attacks. As shown in Figures 6.11 (i) and (iii), offline attacks performed on the traces collected from the vehicles look like spikes because there are multiple random value changes over time that do not follow a linear path. This means that, during an offline attack, the correlation between the vehicle speed and engine speed is reduced because the random changes of vehicle speed and engine speed lead to a minimized relation between the signals. This, of course, would not be the case in a real-vehicle, online, attack, where the correlation between the vehicle speed and engine speed would be maintained also during the attack. This allows us to see the offline attacks as a more synthetic, artificial type, that do not entirely follow the practical behavior of signals on the bus and also diminish the correlation between them. By measuring the correlation coefficients, the correlation in the vehicle trace between vehicle speed is of 0.88 when there is no attack present, 0.70 in case of a fuzzing attack with 10% probability, 0.57 in case of a fuzzing attack with 25% probability, 0.43 in case of a fuzzing attack with 50% probability and 0.35 in case of a fuzzing attack with 75% probability. In case an offline fuzzing attack is performed with a 100% probability, that means attack only, the correlation coefficient for the signals in modified vehicle trace would be 0 because all



(i) Vehicle speed collected from the car

(ii) Vehicle speed computed in the models

(iii) Engine speed collected from the car

(iv) Engine speed computed in the models

Figure 6.11: Vehicle speed and engine speed under a fuzzing attack with 25% probability in an off-line augmented trace (left) and the same signals within the CarTwin models (right)

signal values are random. The CarTwin model has a better correlation between signals during attacks starting from a value of 0.93 when there are no frames attacked. In case of a fuzzing attack with 10% probability the correlation coefficient drops to0.83 while for 25% probability it drops to 0.72. If the fuzzing attack affects 50% or 75% of the legitimate frames and their signal values, the correlation coefficient decreases to 0.61 and 0.56. Even for a fuzzing attack with 100% probability, the correlation coefficient from the CarTwin model is of 0.49. A summary of the experimental results with regards to correlation coefficients is shown in Table 6.4. The results indicate that, even if offline attacks are helpful for intrusion detection systems, they are not aligned to the realistic behavior of automotive systems. On the other hand, the CarTwin model allows the correlation between signals to be preserved, maintaining the signal changes over time as also shown in Figures 6.11 (ii) and (iv). The presumption is the same behavior would be visible in case of attacks on CAN frames and signals in real vehicles. Thus the analysis shows that CarTwin is a good candidate for cyber-attack studies.

Table 6.4: Attack correlation augmented trace vs. CarTwin

	Correlation coefficients					
Experiment	**no attack frames**	**attack at** $p = 0.1$	**attack at** $p = 0.25$	**attack at** $p = 0.5$	**attack at** $p = 0.75$	**attack only**
Generic trace	0.88	0.70	0.57	0.43	0.35	0.00
CarTwin[32]	0.93	0.83	0.72	0.61	0.56	0.49

Another use-case for the CarTwin model is for the safety and fault tolerance studies.

Since the CarTwin model includes the PCM and ABS models which are safety relevant systems from the vehicle that receive or provide safety critical signals such as the vehicle speed, brake, buckle or airbag status, an improvement would include redundant calculations for them using separate input sources or using different computation methods. One practical example is the vehicle speed that can be computed after brake by two distinct models. The ABS model can compute the vehicle speed based on the sensors connected to the brake and the factor of the applied brake and the PCM module can compute the vehicle speed based on acceleration data from a remote sensor. In this way, an attack employed on the vehicle speed provided by the ABS model can be easily observed based on its variation and the variation of the vehicle speed provided by the PCM model. Multiple redundancies of safety critical signals, e.g., 3, 4 or more, may be required to allow a fault tolerant operation of the safety relevant systems.

6.3.5 Comparison with related works

Digital twins for Controller Area Networks are rather new or an emerging topic, so there are just a few research works on this area that can be compared with CarTwin. One of the research works is actually a Masters Thesis [173] that describes the development of a digital twin for vehicle dynamics with the support of the power-steering, braking, and powertrain modules based on MATLAB/Simulink models. Additionally to the utilization of Simulink models, CarTwin also has embedded devices that run the models and that are connected to real-world vehicle bus wiring harnesses that follow the same network topology from the vehicle. The authors from two other works, PASTA (Portable Automotive Security Testbed with Adaptability) [198] and RAMN (Resistant Automotive Miniature Network) [199] propose two automotive testbeds for cybersecurity evaluation that do not have any Simulink models only open source software that can be deployed on the embedded devices. There are various ECU modules in the PASTA testbed that communicate on two separate CAN channels connected through a gateway module. The RAMN testbed is considered small and inexpensive and includes only one CAN channel that allows four ECU modules, i.e., powertrain, chassis, gateway, body control module, to transmit and receive frames. A summary that contains a comparison between CarTwin and related works with respect to the number of modules, usage of Simulink for the models and of real wiring harnesses and topologies is shown in Table 6.5.

Table 6.5: Comparison of research papers addressing Digital Twins for ECUs or automotive testbeds

Research paper	ECUs	Simulink Models	Real-world vehicle bus wiring and topology
PASTA [198]	4	-	-
RAMN [199]	4	-	-
Toyota Digital Twin [173]	3	✓	-
CarTwin [32]	7	✓	✓

6.4 Evaluating the wiring impact on voltage fingerprints using the Digital Twin

Modern cars use the Controller Area Network (CAN) as a main communication channel between Electronic Control Units (ECUs), sensors and actuators. Considering the known vulnerabilities of the CAN bus [2, 45], the automotive industry manufacturers and suppliers are now required to perform vulnerability testing on their products and on the vehicles, as shown by [200, 201]. One of the research areas that cover these vulnerabilities is the physical fingerprinting of nodes that communicate on the Controller Area Networks [27, 65, 127]. There are research works that study voltage fingerprinting methods but use experimental setups that include non-automotive grade wiring [27, 143]. This can have an impact on the resulted datasets due to noise that may be induced by the wiring itself, if improper wiring is used. The newer embodiment of the CAN with Flexible Data Rate (CAN-FD) was standardized and adopted by the automotive industry due to the upcoming needs of autonomous vehicles with regards to the increase of bit rates and frame data payloads [202]. Even though the increase in communication speed is favorable, the voltage characteristic qualities are affected by the fast electrical changes on the CAN-H and CAN-L lines during active communication. Thereby, ringing effects and other signal distortions that may influence voltage signals on CAN-FD networks are studied by authors in [203] for reliable physical bus designs. Electromagnetic emissions can be caused by transmitters, so they have a negative effect on the physical layer of the CAN-FD networks. Authors from [204] study various changes required for CAN-FD transmitters in order to minimize the electromagnetic emission effects during CAN-FD communication. CAN wires are also subject to noise from neighboring equipment, such as actuators or converters. A study from [205] proposes masking the logical flips that may appear during transmission to reduce or eliminate the noise caused by a buck converter. In this way, the electromagnetic interference (EMI) is also studied with respect to the negative influence on the physical layer of Controller Area Networks.

6.4.1 Tools used for data collection and evaluation

The tools that were used to collect the voltage data from the CarTwin [32] experimental setup but also from previous experimental setups from other works [34, 35] are the PicoScope 5000 Series hardware tool and the PicoScope software tool that runs on Windows. This tool allows voltage data to be sampled based on pre-configured triggers and can collect up to 2GS (giga-samples), if one channel is used. If multiple channels are used, depending on the capture window size, the sample rate is reduced to 1GS (giga-sample) or 500MS (mega-samples). In all data collection activities, the PicoScope was configured to capture the voltage data based on a trigger of a rising edge on the CAN-H input that is the SOF (start of frame) bit. Voltage samples for both CAN channels, i.e., CAN-H and CAN-L, are collected during each data collection step. An example of a collected data

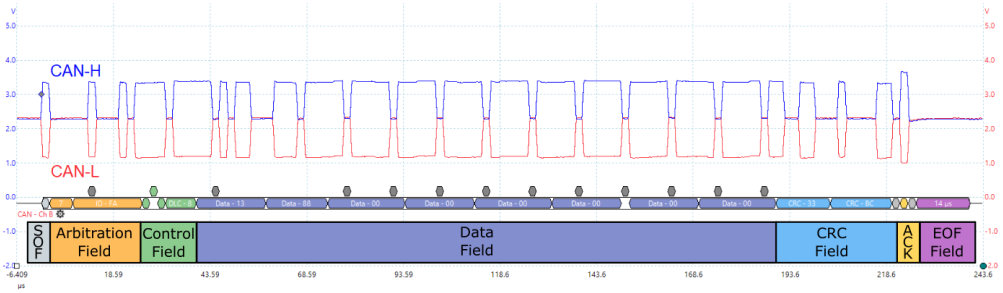


Figure 6.12: CAN-H and CAN-L voltage samples for a CAN frame with details on bit fields

window is shown in Figure 6.12 with a color-coding representation of each bit field from the CAN frame that is reconstructed by the PicoScope software tool based on bit information coming from the voltage data and the selected baud-rate that is of 500Kbps in this case, i.e., bit time is of $2\mu s$. The software tool that was used for voltage data analysis for the various wirings used in the experimental setups is MATLAB R2022a.

6.4.2 Distinct datasets based on various experimental setups

There are several datasets from different experimental setups, built by our group, that are used for voltage analysis. None of these previous setups, with the exception of the aforementioned CarTwin [32], were based on actual in-vehicle wires. Each setup is described in what follows, with information regarding the wiring that is used to define the CAN network including the corresponding AWG (American Wire Gauge) code.

TIDAL-CAN setup. There are 10 devices connected to the CAN network from the experimental setup that is used in TIDAL-CAN [34]. The CAN network from the experimental setup is realized from a cable with three wires, one twisted pair of CAN-H cand CAN-L wires and a ground (GND) wire. The total wire length for the CAN bus is of $5m$ and there are two 120Ω resistors on the left and right ends of the CAN bus. The electrical conductor from the twisted pair wires used for CAN communication is solid bare copper of $0.6mm^2$ (19 AWG). The cable diameter including the insulation part is of $3.1mm$.

CAN-SQUARE setup. There are 5 devices connected to the CAN network in the clean setup from the experimental setup that is used in CAN-SQUARE [35]. In case that adversarial nodes are included, there will be up to 8 devices connected. The CAN network from the experimental setup is realized from an industry grade cable with a twisted pair of wires and a shield as ground. The twisted pair contains the CAN-H and CAN-L wires. The total wire length for the CAN bus is of 5m. There are two $2 \times 60\Omega$ connected to a $10nF$ capacitor, in series, as split terminations, on the left and right ends of the CAN bus. The electrical conductor from the twisted pair wires used for

Table 6.6: Summary of CAN wiring details from experimental setups used in previous works

Experimental Setup	Wire type and its cross-section (AWG)	Wiring	Grade
TIDAL-CAN [34]	Solid Bare Copper Wire of $0.6mm^2$ (19 AWG)	Unshielded Twisted Pair (UTP)	Off-the-shelf
CAN-SQUARE [35]	Stranded Copper Wire of $0.22mm^2$ (24 AWG)	Shielded Twisted Pair (STP)	Industrial
CarTwin [32]	Stranded Copper Wire of $0.5mm^2$ (20 AWG)	Unshielded Twisted Pair (UTP)	Automotive

CAN communication is stranded copper of $0.22mm^2$ (24 AWG). The wire diameter is of $0.6mm$, while the cable diameter including the insulation part is of $5.4mm$.

CarTwin setup. There are 8 devices connected to the CAN network in the clean setup from the experimental setup that is used in CarTwin [32]. The CAN network from the experimental setup is realized from an automotive grade cable from a real-world vehicle with multiple wires including the twisted pair of wires for CAN that was preserved from the original wiring. The twisted pair contains the CAN-H and CAN-L wires. The total wire length excluding stubs and splices for the CAN bus is of $\sim 5m$. The additional stubs have a length that varies from 0.5m to 1.2m, based on the wiring harness location in the real-world vehicle. The 120Ω resistors were preserved on the embedded devices that are close to the end of the CAN bus on the left and right sides and were removed from all the other boards in order to maintain the required impedance for the communication medium. The electrical conductor from the twisted pair wires used for CAN communication is stranded copper of $0.5mm^2$ (20 AWG). The cable diameter including the insulation part is of $2.5mm$.

The information regarding the wires that are used, the wire cross-section and corresponding AWG value, the wiring types and the wiring grade is shown in Table 6.6. An example of the wires of an automotive grade wiring harness from the CarTwin [32] experimental setup is shown in Figure 6.13.

The datasets that are collected from the experimental setups defined in *TIDAL-CAN* [34], *CAN-SQUARE* [35] and *CarTwin* [32] are described in what follows, including the dataset from *ECUPrint* [31] that was collected from real-world vehicles. The reason for including this dataset is to provide comparative results of voltage samples to those collected from the experimental setup configurations.

TIDAL-CAN dataset. There are voltage samples collected from the CAN-H line of 1000 CAN frames, for each experiment performed in the data collection work from *TIDAL-CAN* [34]. The PicoScope tool configuration for voltage sampling is done to collect one voltage sample at every $1ns$ while the collection window size is of $2\mu s$.

CAN-SQUARE dataset. There are voltage samples collected from the CAN-H and CAN-L lines of 500 CAN frames, for each experiment performed in the data collection work from *CAN-SQUARE* [35]. The PicoScope tool configuration for voltage sampling

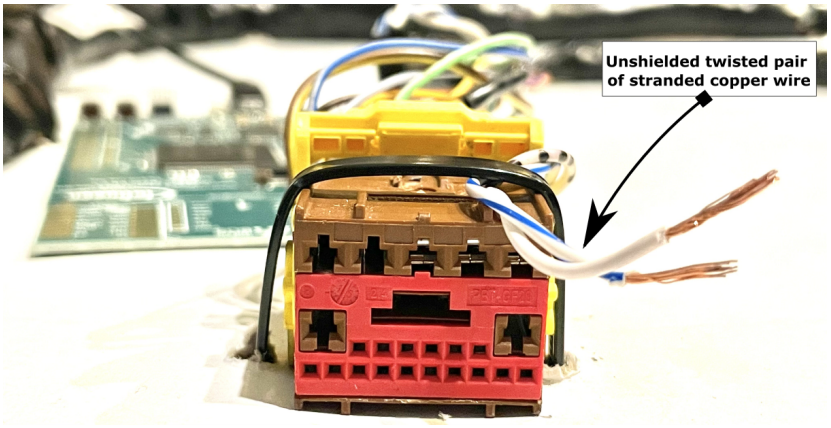


Figure 6.13: Wires from the automotive wiring harness that is part of *CarTwin* [32] experimental setup

is done to collect one voltage sample at every $2ns$ or $4ns$, while the collection window size is of $2\mu s$.

ECUPrint dataset. There are voltage samples collected from the CAN-H and CAN-L lines of multiple frames during the data collection work from *ECUPrint* [31]. The PicoScope tool configuration for voltage sampling is done to collect one voltage sample every $2ns$ while the collection window size is of $2\mu s$. The information from Table 1 in the *ECUPrint* [31] paper contains the total amount of voltage samples collected from each vehicle, e.g., $\sim$32k bits for Dacia Logan.

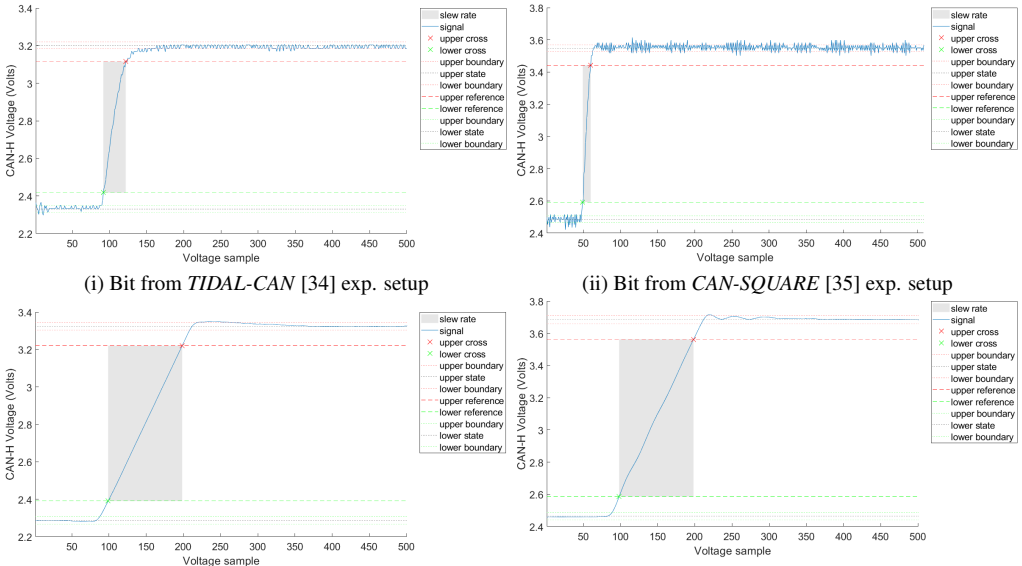
CarTwin dataset. There are voltage samples collected from the CAN-H and CAN-L lines of 916 frames during the data collection work from the *CarTwin* [32] experimental setup done as a contribution in [33]. The PicoScope tool configuration for voltage sampling is done to collect one voltage sample every $2ns$ while the collection window size is of $2\mu s$.

6.4.3 Theoretical framework for data evaluation

Dataset alignment. The first step required before voltage analysis is to align the datasets from previous works since the signals and voltage samples are different for *TIDAL-CAN* [34] and *CAN-SQUARE* [35]. Thereby, only the CAN-H voltage samples are used from each work that contains the rising edge before a dominant bit was transmitted on the bus and part of the plateau of the dominant bit. There are 500 voltage samples used for each CAN-H bit over a time of $2\mu s$ which is the bit time for the 500Kbps baud rate configured for CAN communication in all works. By using this alignment, all voltage data that is used from existing datasets is the same, so the voltage characteristics that are analyzed can be compared.

Slew rate. One of the voltage characteristics that are analyzed is the slew rate which is measured as the absolute voltage change over time. The slew rate can be computed both on rising edges and falling edges and is 0 whenever the voltage is constant. The definition of the measurement of slew rate from the existing datasets is the absolute difference between 10% and 90% of the voltage for CAN-H from the rising edge over the time difference. This is also formalized in Equation 6.9 and visually represented in Figure 6.14. In this figure, the slew rate is emphasized in the gray area, between the green and red dashed lines, for dominant bits, in *TIDAL-CAN* [34] (i) , *CAN-SQUARE* [35] (ii), *CarTwin* [32] (iii) and *ECUPrint* [31] (iv).

$$SR_{CAN} = (|\frac{V_{10\%} - V_{90\%}}{\Delta t}|) \tag{6.9}$$



(i) Bit from *TIDAL-CAN* [34] exp. setup

(ii) Bit from *CAN-SQUARE* [35] exp. setup

(iii) Bit from *CarTwin* [32] exp. setup

(iv) Bit from *ECUPrint* [31] dataset for Dacia Duster vehicle

Figure 6.14: The slew rate represented in MATLAB for the CAN-H rising edge of a bit collected in the *TIDAL-CAN* [34] experimental setup (i), *CAN-SQUARE* [35] experimental setup (ii), *CarTwin* [32] experimental setup (iii) and the *ECUPrint* [31] dataset (iv)

Peak-to-peak and peak to root mean square value. The other voltage characteristics that are analyzed are the peak-to-peak (x_{P2P}) and peak-to-root mean square values (x_{P2RMS}) for the CAN-H voltage. In order to measure the (x_{P2RMS}), the root mean square value is required (x_{RMS}). The peak-to-peak value is computed as a difference between the maximum value and the minimum value of the signal x over a specified number of samples, as shown in Equation 6.10. The root mean square (x_{RMS}) value is computed as the square root of the square sum of N samples, as shown in Equation 6.11. The peak-to-root mean square (x_{P2RMS}) is computed as the ratio between the maximum

value of the signal and its root mean square, as shown in Equation 6.12. An example of the V_{min} , V_{max} and V_{RMS} values computed on 250 voltage samples are shown in the gray area of Figure 6.15 as purple, yellow and orange dashed lines.

$$x_{P2P} = max(x) - min(x) \tag{6.10}$$

$$x_{RMS} = (\sqrt{\frac{1}{N}\sum_{n=1}^{N} x_n^2}) \tag{6.11}$$

$$x_{P2RMS} = \frac{max(x)}{x_{RMS}} \tag{6.12}$$

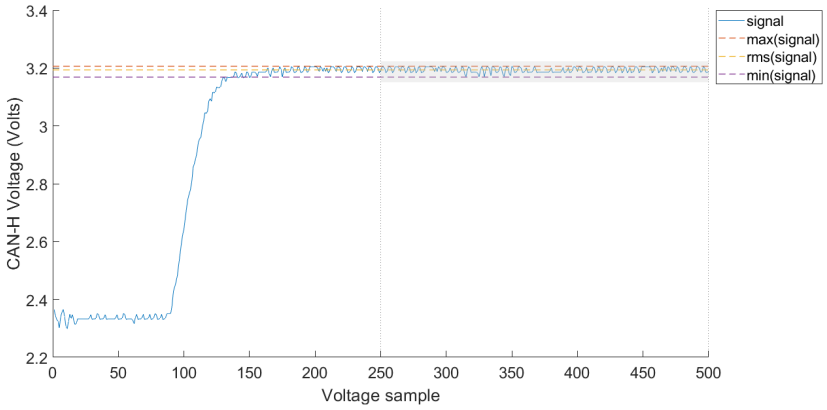
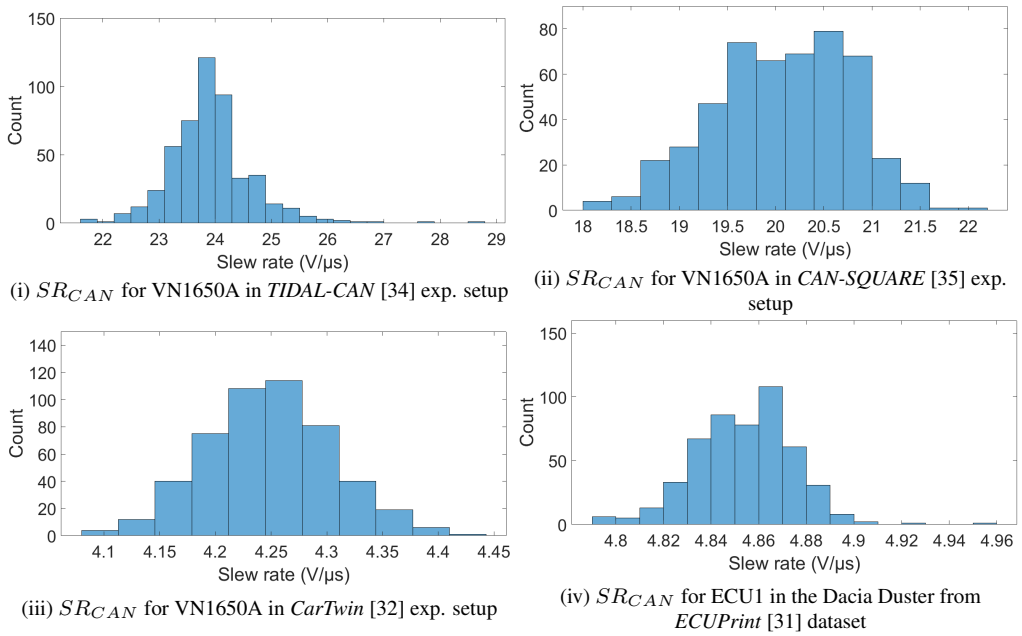


Figure 6.15: The V_{P2P} (V_{min} to V_{max}) and V_{RMS} values represented in MATLAB for the CAN-H dominant plateau area of a bit collected in the *TIDAL-CAN* [34] exp. setup

6.4.4 Data evaluation and discussions

The evaluation is done using 500 bits from each voltage dataset, from CAN frames transmitted by the same VN5610A device. This means a total of 1500 bits that were transmitted by the same device in a CAN network with different wiring. In order to compare the results with those from real-world vehicles, there are 500 bits used from the ECUPrint [31] dataset for the Dacia Duster passenger car. The voltage characteristics which are analyzed in what follows are the slew rate for the CAN-H line transition from recessive to dominant state and the peak-to-peak and peak to root mean square values from the plateau area of the dominant bit.

Slew rate distribution. As already described with the theoretical framework, the slew rate is computed between 10% and 90% of the voltage range for the rising edge of a dominant bit on the CAN-H line. The values that are obtained for the TIDAL-CAN [34] distributions are between $21V/\mu s$ and $29V/\mu s$ with most of the values between



(i) SR_{CAN} for VN1650A in *TIDAL-CAN* [34] exp. setup

(ii) SR_{CAN} for VN1650A in *CAN-SQUARE* [35] exp. setup

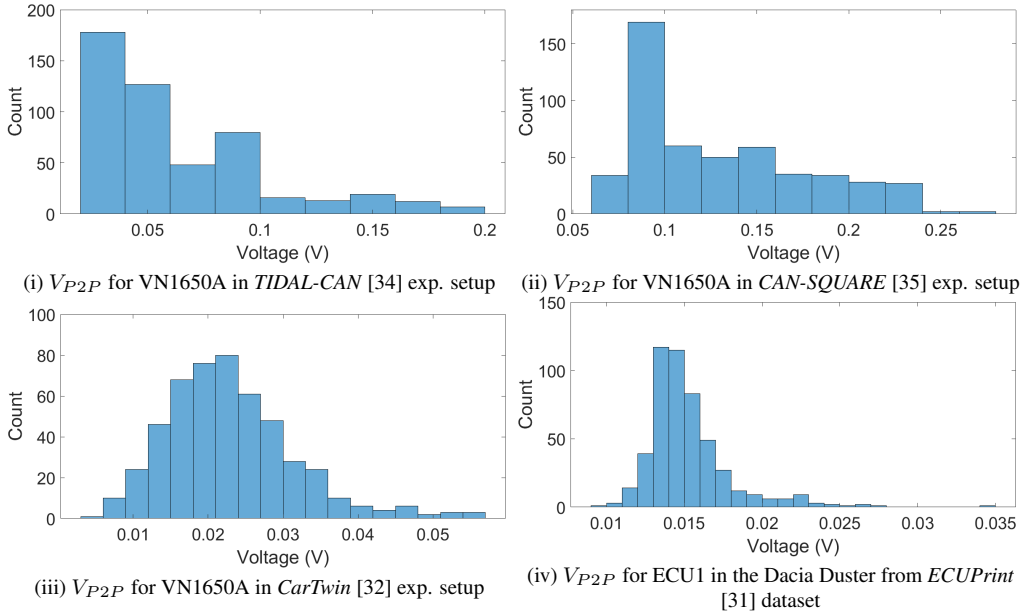
(iii) SR_{CAN} for VN1650A in *CarTwin* [32] exp. setup

(iv) SR_{CAN} for ECU1 in the Dacia Duster from *ECUPrint* [31] dataset

Figure 6.16: The SR_{CAN} (slew rate) distribution for 500 bits transmitted by Vector VN5610A in the *TIDAL-CAN* [34] experimental setup (i), *CAN-SQUARE* [35] experimental setup (ii), *CarTwin* [32] experimental setup (iii) and by ECU1 in the Dacia Duster from the *ECUPrint* [31] dataset (iv)

$23V/\mu s$ and $25V/\mu s$. This is also shown in Figure 6.16 (i). For the CAN-SQUARE [35] dataset, the slew rate values are between $18V/\mu s$ and $22V/\mu s$ with most of them in the $19V/\mu s - 21V/\mu s$ range. This is presented in Figure 6.16 (ii). For the CarTwin [32] dataset, the slew rate values are between $4.10V/\mu s$ and $4.45V/\mu s$ with most of the values between $4.2V/\mu s$ and $4.3V/\mu s$. This is presented in Figure 6.16 (iii). For comparative purposes with real-world values, the slew rate distribution obtained from the ECUPrint [31] dataset, contains values in the $4.7 - 5V/\mu s$ range. This is visually depicted in Figure 6.16 (iv). By comparing the values obtained from the experimental setup datasets with those from the real-world vehicle dataset, there is a clear difference in the slew rate that depends on the wire type and the wiring used. The slew rate distribution for compared values from the CarTwin [32] and ECUPrint [31] datasets has values below $10V/\mu s$. This means that the slew rate characteristic for the CAN bus from CarTwin [32] is very close to the real conditions from the vehicle. This is somehow expected because the CAN bus was preserved from the original wiring harnesses that were removed from a real vehicle.

Peak-to-peak distribution. As already described in the theoretical framework and shown in the gray area of Figure 6.15, the peak-to-peak value is computed over 250 voltage samples from the plateau area of a dominant bit on the CAN-H line. The values that are obtained for the TIDAL-CAN [34] distributions are between $25mV$ and $200mV$



(i) V_{P2P} for VN1650A in *TIDAL-CAN* [34] exp. setup

(ii) V_{P2P} for VN1650A in *CAN-SQUARE* [35] exp. setup

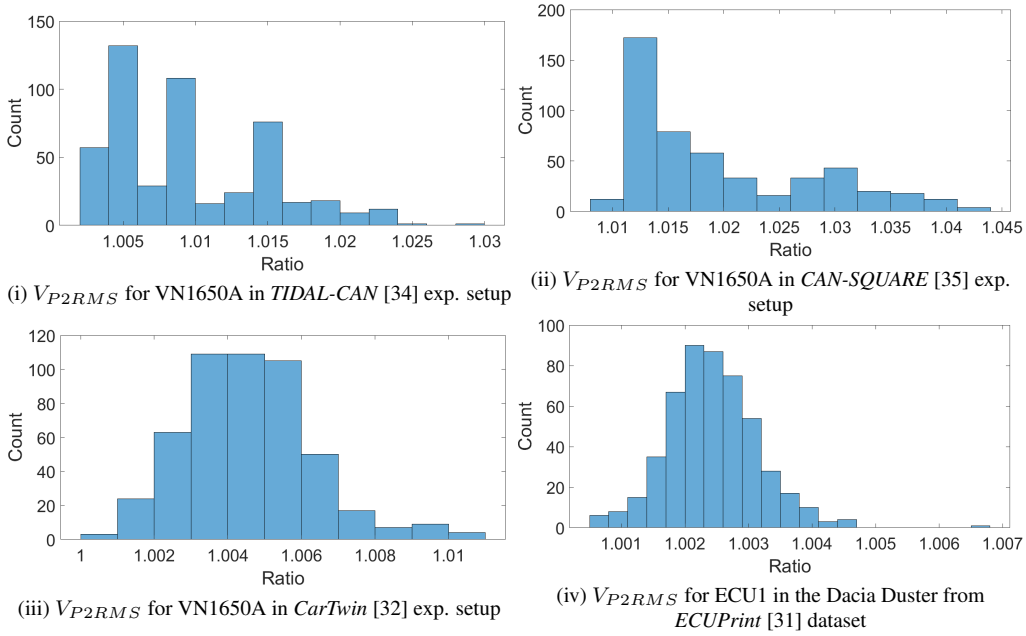
(iii) V_{P2P} for VN1650A in *CarTwin* [32] exp. setup

(iv) V_{P2P} for ECU1 in the Dacia Duster from *ECUPrint* [31] dataset

Figure 6.17: The V_{P2P} (voltage peak-to-peak) distribution for 500 bits transmitted by Vector VN5610A in the *TIDAL-CAN* [34] experimental setup (i), *CAN-SQUARE* [35] experimental setup (ii), *CarTwin* [32] experimental setup (iii) and by ECU1 in the Dacia Duster from the *ECUPrint* [31] dataset (iv)

with most of the values between $25mV$ and $100mV$. This is also shown in Figure 6.17 (i). For the CAN-SQUARE [35] dataset, the slew rate values are between $50mV$ and $275mV$ with most of them in the $50mV - 150mV$ range. This is presented in Figure 6.17 (ii). For the CarTwin [32] dataset, the slew rate values are between 5mV and 55mV. This is presented in Figure 6.17 (iii). For comparative purposes with real-world values, the peak-to-peak distribution obtained from the ECUPrint [31] dataset, contains values in the $9mV - 35mV$ range. This is visually depicted in Figure 6.17 (iv). By comparing the values obtained from the experimental setup datasets with those from the real-world vehicle dataset, there is a higher value for the peak-to-peak voltage on the plateau area of a bit that may be influenced by the wire cross-section that can cause additional noise. The peak-to-peak distributions from the CarTwin [32] and ECUPrint [31] datasets have values below $60mV$. This means that the peak-to-peak voltage distribution for the CAN bus from CarTwin [32] is very close to the real conditions from the vehicle.

Peak-to-RMS distribution. As already described within the theoretical framework, the peak-to-RMS value is computed over 250 voltage samples from the plateau area of a dominant bit on the CAN-H line. The values that are obtained for the TIDAL-CAN [34] distributions are between 1.004 and 1.030, which means $max(V)$ is higher by up to 3% compared to V_{RMS}. This is also shown in Figure 6.18 (i). For the CAN-SQUARE [35] dataset, the peak-to-RMS values are between 1.010 and 1.040, which means $max(V)$ is



(i) V_{P2RMS} for VN1650A in *TIDAL-CAN* [34] exp. setup

(ii) V_{P2RMS} for VN1650A in *CAN-SQUARE* [35] exp. setup

(iii) V_{P2RMS} for VN1650A in *CarTwin* [32] exp. setup

(iv) V_{P2RMS} for ECU1 in the Dacia Duster from *ECUPrint* [31] dataset

Figure 6.18: The V_{P2RMS} (voltage peak-to-RMS) distribution for 500 bits transmitted by Vector VN5610A in the *TIDAL-CAN* [34] experimental setup (i), *CAN-SQUARE* [35] experimental setup (ii), *CarTwin* [32] experimental setup (iii) and by ECU1 in the Dacia Duster from the *ECUPrint* [31] dataset (iv)

higher by up to 4% than V_{RMS}. This is presented in Figure 6.18 (ii). For the CarTwin [32] dataset, the peak-to-RMS values are between 1.000 to 1.011. This is presented in Figure 6.18 (iii). For comparative purposes with real-world values, the peak-to-RMS distribution obtained from the ECUPrint [31] dataset, contains values in the $1.001 - 1.007$ range. This is visually depicted in Figure 6.18 (iv). By comparing the values obtained from the experimental setup datasets with those from the real-world vehicle dataset, there is a higher value for the peak-to-RMS voltage on the plateau area of a bit that may be influenced by the wire characteristics in a similar way it does for the peak-to-peak. The peak-to-RMS distributions from the CarTwin [32] and ECUPrint [31] datasets have values below 1.02. This means that the peak-to-RMS voltage distribution for the CAN bus from CarTwin [32] is very close to the real conditions from the vehicle.

6.5 Concluding remarks

In this chapter, the digital twin of a real CAN vehicle network, i.e., CarTwin, was presented. There are 7 MATLAB/Simulink models which were described for the systems connected on a CAN bus from a vehicle. Moreover, the original CAN wires from three

wiring harnesses which were removed from a real-world vehicle, define the CAN bus for CarTwin. Since the models require additional vehicle signals, a tool designed in C# that provides the expected inputs was described. The deployment of the models on automotive-grade embedded boards was also presented, with details related to the execution cycle time, software drivers and timer configurations. Three of the CarTwin model output signals, i.e., vehicle speed, engine speed and trip distance, were compared to those from a real car during local road driving and highway driving conditions. Further, the wiring impact on CAN voltage fingerprints was discussed based on the car twin that we designed. Three experimental setups were described with details regarding the wire conductor type, wiring type and grade. The voltage datasets collected from these experimental setups have been presented together with an additional voltage dataset collected from real-world vehicles, that was used for comparison. The voltage characteristics that were analyzed from the datasets are the slew rate for CAN-H rising edges of dominant bits, peak-to-peak and peak-to-root mean square on the plateau area of dominant bits. The values obtained for dominant bits from the CarTwin experimental setup dataset are closer to those from the real-world vehicle dataset than those obtained for the other experimental setups. This shows that the selection of the wire type and wiring harness is an important factor for voltage-based fingerprinting in Controller Area Networks.

Chapter 7

Conclusion

This thesis examined various security applications and their evaluation for Controller Area Networks. The security applications utilize the physical layer for cryptographic key-exchange, sender and frame authentication and transmitter fingerprinting. These applications were either compared with existing proposals by emphasizing the performance improvements or analyzed in the context of existing related works from the literature.

Chapter 3 presents time-covert key exchange mechanism for the Controller Area Networks. The chapter begins with a presentation of the computational cost with regards to timing if only software-based elliptic curve cryptography is utilized for key exchange or digital signatures on Controller Area Networks. Evaluation of software execution time for the Elliptic Curve Diffie-Hellmann (ECDH) key exchange and Elliptic Curve Digital Signature Algorithms (ECDSA) is performed on an automotive grade microcontroller from the AURIX family produced by Infineon. The experimental results are compared between three open-source libraries which provide the implementation of the primitives for the evaluated elliptic curve security algorithms. Nevertheless, the ECDH key exchange requires multiple CAN frames to be transmitted until the key negotiation is performed. In what follows from the same chapter, a time-covert key-exchange protocol between two CAN nodes is described, after the background and related studies are discussed. The evaluation platform used for the protocol evaluation is also presented before the protocol. The key-exchange protocol has four different key exchange methods proposed. The first one is Data vs. Remote frame negotiation which is based on the arbitration procedure on the physical layer of the CAN bus between randomly transmitted as data frame or remote frames, with the same identifier, at fixed time intervals. The next method is Minimax Negotiation that is also based on the arbitration procedure but requires data frames to be transmitted with random identifiers, at fixed time slots. The following method is the Time-Triggered Minimax Negotiation which is based on both random identifiers and random time slots inside the same time interval. The last method is the Randomized Time-Triggered Key Exchange that is based on a specified number of frames and random time slots when these frames are transmitted. All key exchange

methods are compared with respect to three characteristics. These are the probability that a frame can be used for extraction of a bit from the session key, the mean entropy and the time it takes to execute the method. Since the shared secret resulted from the key exchange has a lower entropy than the key normally used in cryptographic protocols, an extension of the last two methods is proposed using the Simple Password Exponential Key Exchange (SPEKE) protocol. The multi-party version of the proposed methods or their extension is described at the end of this chapter.

Chapter 4 presents a time-covert authentication protocol on the Controller Area Networks that is improved through frame scheduling optimization. In the first part of the chapter, the background and related studies are discussed together with limitations from existing works that are overcome with the proposal discussed in the next sections. The worst-case arrival times for CAN frames are discussed as a theoretical background that is practically evaluated using the frame arrival times from real vehicle times. Considering the delays that one CAN frame may have from the expected transmission time to the time it is actually sent and in the end received by the other nodes, the frame scheduling optimization is proposed to overcome these findings. As input for the frame scheduling optimization, there is a dataset defined that contains all the frame identifiers, their cycle times and the time offset that is added to the cycle time for each frame identifier. The evaluation platform which consists of an automotive device for CAN communication and an embedded device is described before the algorithms are discussed. There are four scheduling optimization algorithms which are described as part of this chapter. The first one is the Binary Symmetric Allocation algorithm which is simple to determine and integrate but has some timing issues in the practical implementation compared to the theoretical distribution caused by small inter-frame spaces. The second is the Randomized Search Allocation algorithm that, even though it has an increased inter-frame space compared to the first algorithm, it still has timing issues in the practical implementation. The next one is the Greedy Allocation algorithm which is described both in single layer and multi-layer variants. The Multi-Layer Greedy variant allows an inter-frame space that is two times the one from the single-layer Greedy Allocation or the Randomized Search Allocation. The experimental results for Multi-Layer Greedy implementation are those expected from the theoretical distribution. The last algorithm is the GCD-based Allocation which allows the same inter-frame space as the Multi-Layer Greedy Allocation. Thereby, its practical implementation follows the theoretical distribution with no deviations. Each algorithm is evaluated based on the minimization of a quality factor that depends on the optimization of inter-frame spacing and the minimum value of the inter-frame space. After the frame scheduling optimization algorithms are described, the time-covert authentication protocol from a previous work is presented together with the adversary model that is considered for its evaluation. The evaluation of the time-covert authentication protocol is done in different scenarios. The first scenario is based on optimized traffic and a single sender. The next scenario is based on both optimized and unoptimized traffic with multiple senders. In the case of multiple senders, there are

additional actions proposed in order to improve the performance of the time-covert channel. These are the re-synchronization of the transmitters time and the de-skewing of their clocks. The channel data rate and security level of the time-covert authentication protocol is described at the end of the second section. The final section of this chapter presents the comparison of the time-covert channel with frame scheduling optimization with related studies.

Chapter 5 presents a physical fingerprinting approach for Electronic Control Units (ECUs) that communicate on the Controller Area Networks inside real-world vehicles. The first section of this chapter contains details related to existing works that perform either timing-based or voltage-based fingerprinting for nodes connected on CAN buses. The summary of the data collection framework, the data collected by the thesis author as well as a comparison between clock skews and voltage features for ECU fingerprinting are also shown as part of this section. The following section details the theoretical framework for computing clock skews and voltage features that are extracted from the sample data that is collected from several passenger vehicles. There are four voltage features which are determined for each frame identifier. The voltage features that are used in what follows are the mean and maximum voltages from the plateau area of a dominant bit, the bit time and the plateau time. For the separation of IDs from the same ECU or from different ECUs, the intra-distances and inter-distances are also defined using the Euclidian distance. The final section of this chapter details the values that are obtained for each passenger vehicle and emphasizes the limitations in case only one physical characteristic is used. The separation is done based on the feature values determined from the data collected for each frame identifier. The frame identifiers are separated into ECUs based on the voltage features and the resulted clock skews. There are 51 ECUs identified using physical characteristics from 9 passenger vehicles based on data collected for close to 400 frame identifiers. For some vehicles such as the Ford Fiesta and Ford Kuga, the clock skew determination was somehow problematic since the variation of reception time was inconsistent. In the dataset collected for Dacia Logan, there are two ECUs separated based on voltage features which have a clock skew difference of only 1ppm (part per million). The inter-distances and intra-distances show separations between IDs from the same ECU or from different ECUs but there are multiple collisions in case only one physical feature is used. When all voltage features are combined, the number of collisions is reduced and the separation becomes very clear. The environmental influence on physical fingerprinting is the last topic discussed in this chapter. Based on data collected after startup and after 1 hour drive from two vehicles, both timing and voltage-based characteristics are evaluated with respect to changes over time. Since the changes are both positive and negative meaning that the values have either increased or decreased over time, the environmental impact on physical fingerprinting cannot be generalized. This is why, updates of fingerprints need to be performed in order to keep the collected values as close as possible to the expected data.

Chapter 6 presents a digital twin for a real-world vehicle Controller Area Network.

The first section of this chapter starts with the presentation of the motivation for the design of the digital twin for a CAN bus as well as the related studies. The digital twin is based on ECU models deployed on automotive grade boards that are physically connected to a CAN bus. The CAN bus is preserved from existing wiring harnesses that were removed from a real-world vehicle. The Electronic Control Units (ECUs) from the real-world vehicle connected to the physical CAN bus were determined from a wiring handbook diagram. The physical wiring from the real-world vehicle is also described with respect to the wiring length and number of stubs. After the physical medium for the CAN bus is described, the design and validation aspects for the ECU models that were performed in MATLAB/Simulink are presented. The ECUs which are modeled and described are the Accessory Protocol Interface Module (APIM), Power Steering Control Module (PSCM), Instrument Panel Cluster (IPC), Remote Function Actuator (RFA), Restraints Control Module (RCM), Anti-lock Brake System (ABS) and Powertrain Control Module (PCM). The ECU models were afterwards deployed on automotive-grade microcontrollers from the AURIX family that are produced by Infineon. The development of a C#-based tool that provides specific vehicle level signals required by some of the ECU models is also described in this section. The integration of the models on the embedded devices was done by verifying the outputs from the CAN bus with those from the modelling tool having the same input arrays provided for both. Afterwards, the vehicle speed and engine speed provided by the digital twin models are compared with the vehicle speed and engine speed from a real world vehicle. The input that is provided to the digital twin model is the brake status that reduces the vehicle speed and engine speed, if applied. There are two driving conditions which are compared, the local roads and the highway. A statistical comparison is performed by showing the differences and correlation coefficients for vehicle speed from the model and the vehicle trace. Possible applications for the digital twin model as well as a brief comparison with related works are shown in what follows. The second section addresses the wiring impacts on voltage fingerprints performed on Controller Area Networks. Several related works are presented before the data collection tool configuration is described. The wiring impacts are evaluated using datasets from three experimental setups with different wirings and a dataset collected from a real-world vehicle. The characteristics which are evaluated are the slew rate for a rising edge, for a dominant bit, the peak-to-peak value on the bit plateau area and the peak-to-root mean square value on the same area. Considering the differences between the wirings used in experimental setups and the real-world vehicle, it seems that automotive cables are highly recommended to be used for testbeds where the intention is to collect voltage samples for fingerprinting the nodes that communicate on the CAN bus. If other wiring types are used, additional noise that may be induced by the cables would have a negative impact on the fingerprinting results.

To summarize, this thesis presents various methods for securing the Controller Area Network that can be implemented on automotive-grade embedded devices, from time-covert authentication protocols, which benefit from frame scheduling optimization, to

physical device fingerprinting using time and voltage characteristics. An experimental setup which is realized as a digital twin for a real-world vehicle Controller Area Network is also described. The wiring impact from this experimental setup is evaluated in the context of voltage-based fingerprinting, a relevant topic for transmitter identification and intrusion detection in Controller Area Networks. There are still many open questions regarding the use of physical layer security on CAN buses which can serve as future works. With regards to time-covert authentication channels, an open research topic is related to the maximum data-rate that can be retrieved from such channels. If voltage or timing characteristics are used as fingerprints for CAN transmitters, their stability over time and the impact of the ECUs' voltage supply are future research directions. Since digital twins are an emerging topic in the automotive areas, they can be further used for studies of CAN safety and cybersecurity in the automotive context. Automotive digital twins clearly need much more exploration at the current time since only a few papers about them were published so far.

List of Figures

List of Tables

Bibliography

[1] 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 *2010 IEEE Symposium on Security and Privacy*. IEEE, 2010, pp. 447–462.

[2] 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.

[3] *Specification of Secure Onboard Communication*, 4.2.2 ed., AUTOSAR, 2014.

[4] M. Wille, "Automotive security—an overview of standardization in AUTOSAR," *VDI/VW-Gemeinschaftstagung Automotive Security*, 2015.

[5] A. Hazem and H. Fahmy, "LCAP - A Lightweight CAN Authentication Protocol for Securing In-Vehicle Networks," in *10th escar Embedded Security in Cars Conference, Berlin, Germany*, vol. 6, 2012, p. 172.

[6] B. Groza, P.-S. Murvay, A. Van Herrewege, and I. Verbauwhede, "LiBrA-CAN: a Lightweight Broadcast Authentication protocol for Controller Area Networks," in *11th International Conference on Cryptology and Network Security, CANS 2012, Springer-Verlag, LNCS*, 2012.

[7] A.-I. Radu and F. D. Garcia, "LeiA: a lightweight authentication protocol for CAN," in *European Symposium on Research in Computer Security*. Springer, 2016, pp. 283–300.

[8] B. Groza, L. Popa, and P.-S. Murvay, "Highly Efficient Authentication for CAN by Identifier Reallocation with Ordered CMACs," *IEEE Transactions on Vehicular Technology*, vol. 69, no. 6, pp. 6129–6140, 2020.

[9] K. Han, A. Weimerskirch, and K. G. Shin, "A practical solution to achieve real-time performance in the automotive network by randomizing frame identifier," *Proc. Eur. Embedded Secur. Cars (ESCAR)*, pp. 13–29, 2015.

[10] A. Humayed and B. Luo, "Using ID-hopping to defend against targeted DoS on CAN," in *Proceedings of the 1st International Workshop on Safe Control of Connected and Autonomous Vehicles*. ACM, 2017, pp. 19–26.

[11] S. Woo, D. Moon, T.-Y. Youn, Y. Lee, and Y. Kim, "CAN ID Shuffling Technique (CIST): Moving Target Defense Strategy for Protecting In-Vehicle CAN," *IEEE Access*, vol. 7, pp. 15 521–15 536, 2019.

[12] "CAN Injection: keyless car theft," https://kentindell.github.io/2023/04/03/can-injection/, 2023, accessed: 2023-05-12.

[13] "CVE-2023-29389," https://nvd.nist.gov/vuln/detail/CVE-2023-29389, 2023, accessed: 2023-05-12.

[14] "CVE-2017-14937," https://cve.mitre.org/cgi-bin/cvename.cgi?name=CVE-2017-14937, 2017, accessed: 2023-05-12.

[15] "CVE-2018-9322," https://cve.mitre.org/cgi-bin/cvename.cgi?name=CVE-2018-9322, 2018, accessed: 2023-05-12.

[16] H. Wen, Q. A. Chen, and Z. Lin, "Plug-N-pwned: Comprehensive vulnerability analysis of OBD-II dongles as a new over-the-air attack surface in automotive IoT," in *USENIX Security Symposium*, 2020.

[17] H. Wen, Q. Zhao, Q. A. Chen, and Z. Lin, "Automated cross-platform reverse engineering of CAN bus commands from mobile apps," in *Proceedings 2020 Network and Distributed System Security Symposium (NDSS'20)*, 2020.

[18] J. Van den Herrewegen and F. D. Garcia, "Beneath the bonnet: A breakdown of diagnostic security," in *European Symposium on Research in Computer Security*. Springer, 2018, pp. 305–324.

[19] S. Nie, L. Liu, and Y. Du, "Free-fall: Hacking Tesla from wireless to CAN bus," *Briefing, Black Hat USA*, vol. 25, pp. 1–16, 2017.

[20] D. S. Fowler, J. Bryans, S. A. Shaikh, and P. Wooderson, "Fuzz testing for automotive cyber-security," in *2018 48th Annual IEEE/IFIP International Conference on Dependable Systems and Networks Workshops (DSN-W)*. IEEE, 2018, pp. 239–246.

[21] Y. Wang, J. An, and Y. Ha, "Unified data authenticated encryption for vehicular communication," in *2016 IEEE 59th International Midwest Symposium on Circuits and Systems (MWSCAS)*. IEEE, 2016, pp. 1–4.

[22] S. FTAIMI and T. MAZRI, "Risk Assessment of Attack in Autonomous Vehicle based on a Decision Tree," *International Journal of Advanced Computer Science and Applications*, vol. 12, no. 7, 2021.

[23] H. M. Song, H. R. Kim, and H. K. Kim, "Intrusion detection system based on the analysis of time intervals of CAN messages for in-vehicle network," in *2016 international conference on information networking (ICOIN)*. IEEE, 2016, pp. 63–68.

[24] B. Groza, L. Popa, and P.-S. Murvay, "INCANTA – Intrusion detection in Controller Area Networks with Time-covert Cryptographic Authentication," in *Security and Safety Interplay of Intelligent Software Systems: ESORICS 2018 International Workshops, ISSA 2018 and CSITS 2018, Barcelona, Spain, September 6–7, 2018, Revised Selected Papers*. Springer, 2019, pp. 94–110.

[25] P.-S. Murvay, L. Popa, and B. Groza, "Securing the Controller Area Network with covert voltage channels," *International Journal of Information Security*, vol. 20, no. 6, pp. 817–831, 2021.

[26] P.-S. Murvay and B. Groza, "Source Identification Using Signal Characteristics in Controller Area Networks," *IEEE Signal Processing Letters*, vol. 21, no. 4, pp. 395–399, 2014.

[27] K.-T. Cho and K. G. Shin, "Viden: Attacker Identification on In-Vehicle Networks," in *Proceedings of the 2017 ACM SIGSAC Conference on Computer and Communications Security*, 2017, pp. 1109–1123.

[28] L. Popa, B. Groza, and P.-S. Murvay, "Performance Evaluation of Elliptic Curve Libraries on Automotive-Grade Microcontrollers," in *Proceedings of the 14th International Conference on Availability, Reliability and Security*, 2019, pp. 1–7.

[29] B. Groza, L. Popa, and P.-S. Murvay, "TRICKS—Time TRIggered Covert Key Sharing for Controller Area Networks," *IEEE Access*, vol. 7, pp. 104 294–104 307, 2019.

[30] ——, "CANTO-Covert AutheNtication with Timing channels over Optimized traffic flows for CAN," *IEEE Transactions on Information Forensics and Security*, vol. 16, pp. 601–616, 2020.

[31] L. Popa, B. Groza, C. Jichici, and P.-S. Murvay, "ECUPrint—Physical Fingerprinting Electronic Control Units on CAN Buses Inside Cars and SAE J1939 Compliant Vehicles," *IEEE Transactions on Information Forensics and Security*, vol. 17, pp. 1185–1200, 2022.

[32] L. Popa, A. Berdich, and B. Groza, "CarTwin—Development of a digital twin for a real-world in-vehicle CAN network," *Applied Sciences*, vol. 13, no. 1, p. 445, 2022.

[33] L. Popa, C. Jichici, T. Andreica, P.-S. Murvay, and B. Groza, "Impact of Wiring Characteristics on Voltage-based Fingerprinting in Controller Area Networks," in *IEEE 17th International Symposium on Applied Computational Intelligence and Informatics (SACI)*, 2023, pp. 231–236.

[34] P.-S. Murvay and B. Groza, "TIDAL-CAN: Differential Timing based Intrusion Detection And Localization for Controller Area Network," *IEEE Access*, vol. 8, pp. 68 895–68 912, 2020.

[35] B. Groza, P.-S. Murvay, L. Popa, and C. Jichici, "CAN-SQUARE - Decimeter Level Localization of Electronic Control Units on CAN Buses," in *Computer Security–ESORICS 2021: 26th European Symposium on Research in Computer Security, Darmstadt, Germany, October 4–8, 2021, Proceedings, Part I 26*. Springer, 2021, pp. 668–690.

[36] F. Consortium *et al.*, "Flexray communications system protocol specification version 2.1," 2005.

[37] P.-S. Murvay, L. Popa, and B. Groza, "Accommodating Time-Triggered Authentication to FlexRay Demands," in *Proceedings of the Third Central European Cybersecurity Conference*, 2019, pp. 1–6.

[38] A. Perrig, R. Canetti, D. Song, P. D. Tygar, and B. Briscoe, "Timed Efficient Stream Loss-Tolerant Authentication (TESLA): Multicast Source Authentication Transform Introduction," RFC 4082, Jun. 2005. [Online]. Available: https://www.rfc-editor.org/info/rfc4082

[39] B. Groza, L. Popa, and P.-S. Murvay, "CarINA-Car sharing with IdeNtity based Access control re-enforced by TPM," in *Computer Safety, Reliability, and Security: SAFECOMP 2019 Workshops, ASSURE, DECSoS, SASSUR, STRIVE, and WAISE, Turku, Finland, September 10, 2019, Proceedings 38*. Springer, 2019, pp. 210–222.

[40] B. Groza, H. Gurban, L. Popa, A. Berdich, and S. Murvay, "Car-to-Smartphone Interactions: Experimental Setup, Risk Analysis and Security Technologies," in *5th International Workshop on Critical Automotive Applications: Robustness & Safety*, 2019.

[41] A. Musuroi, B. Groza, L. Popa, and P.-S. Murvay, "Fast and Efficient Group Key Exchange in Controller Area Networks (CAN)," *IEEE Transactions on Vehicular Technology*, vol. 70, no. 9, pp. 9385–9399, 2021.

[42] B. Groza, L. Popa, P.-S. Murvay, Y. Elovici, and A. Shabtai, "CANARY-a reactive defense mechanism for Controller Area Networks based on Active RelaYs." in *USENIX Security Symposium*, 2021, pp. 4259–4276.

[43] B. Groza, L. Popa, T. Andreica, P.-S. Murvay, A. Shabtai, and Y. Elovici, "PanoptiCANs-Adversary-Resilient Architectures for Controller Area Networks," in *Computer Security–ESORICS 2022: 27th European Symposium on Research in Computer Security, Copenhagen, Denmark, September 26–30, 2022, Proceedings, Part III*. Springer, 2022, pp. 658–679.

[44] ISO, "ISO11898-1. Road vehicles — Controller Area Network (CAN) — Part 1: Data link layer and physical signalling," International Organization for Standardization, Standard, 2nd edition, Dec 2015.

[45] C. Miller and C. Valasek, "A survey of remote automotive attack surfaces," *Black Hat USA*, 2014.

[46] *Specification of Crypto Driver*, R22-11 ed., AUTOSAR, 2022.

[47] *Specification of Crypto Service Manager*, R22-11 ed., AUTOSAR, 2022.

[48] M. Ltd., "Multiprecision Integer and Rational Arithmetic C Library – the MIRACL Crypto SDK," 2019, accessed: 2023-05-28. [Online]. Available: https://github.com/miracl/MIRACL

[49] D. F. Aranha and C. P. L. Gouvêa, "RELIC is an Efficient LIbrary for Cryptography," 2019, accessed: 2023-05-28. [Online]. Available: https://github.com/relic-toolkit/relic

[50] D. F. Pigatto, N. B. F. da Silva, and K. R. L. J. C. Branco, "Performance evaluation and comparison of algorithms for elliptic curve cryptography with El-Gamal based on MIRACL and RELIC libraries," *Journal of Applied Computing Research*, vol. 1, no. 2, pp. 95–103, 2011. [Online]. Available: http://revistas.unisinos.br/index.php/jacr/article/view/jacr.2011.12.04/675

[51] R. de Clercq, L. Uhsadel, A. V. Herrewege, and I. Verbauwhede, "Ultra Low-Power Implementation of ECC on the ARM Cortex-M0+," in *Proceedings of the 51st Annual Design Automation Conference*, ser. DAC '14. New York, NY, USA: ACM, 2014, pp. 112:1–112:6. [Online]. Available: http://doi.acm.org/10.1145/2593069.2593238

[52] G. Hinterwälder, A. Moradi, M. Hutter, P. Schwabe, and C. Paar, "Full-size high-security ECC implementation on MSP430 microcontrollers," in *International Conference on Cryptology and Information Security in Latin America*. Springer, 2014, pp. 31–47. [Online]. Available: https://doi.org/10.1007/978-3-319-16295-9_2

[53] SECG, Elliptic Curve Cryptography, "Standards for efficient cryptographic group."

[54] P.-S. Murvay, A. Matei, C. Solomon, and B. Groza, "Development of an AUTOSAR Compliant Cryptographic Library on State-of-the-Art Automotive Grade Controllers," in *2016 11th International Conference on Availability, Reliability and Security (ARES)*. IEEE, 2016, pp. 117–126.

[55] D. Johnson, A. Menezes, and S. Vanstone, "The elliptic curve digital signature algorithm (ECDSA)," *International journal of information security*, vol. 1, pp. 36–63, 2001.

[56] D. Zelle, C. Krauß, H. Strauß, and S. Karsten, "On Using TLS to Secure In-Vehicle Networks," in *ARES '17 Proceedings of the 12th International Conference on Availability, Reliability and Security, Article No. 67*. ACM, 2017. [Online]. Available: https://doi.org/10.1145/3098954.3105824

[57] wolfSSL Inc., "wolfSSL embedded SSL library," 2019, accessed: 2023-05-28. [Online]. Available: https://www.wolfssl.com/products/wolfssl/

[58] R. L. Rivest, A. Shamir, and L. Adleman, "A method for obtaining digital signatures and public-key cryptosystems," *Communications of the ACM*, vol. 21, no. 2, pp. 120–126, 1978.

[59] wolfSSL Inc., "wolfCrypt Embedded Crypto Engine," 2019, accessed: 2023-05-28. [Online]. Available: https://www.wolfssl.com/products/wolfcrypt/

[60] J.-P. Aumasson, L. Henzen, W. Meier, and R. C.-W. Phan, "SHA-3 proposal BLAKE," *Submission to NIST*, vol. 229, p. 230, 2008.

[61] N. I. of Standards and Technology, "FIPS 140-2. Security Requirements for Cryptographic Modules," National Institute of Standards and Technology, Standard, 1st edition, May 2001.

[62] B. Groza and P.-S. Murvay, "Security Solutions for the Controller Area Network: Bringing Authentication to In-Vehicle Networks," *IEEE Vehicular Technology Magazine*, vol. 13, no. 1, pp. 40–47, 2018.

[63] O. Hartkopp, C. Reuber, and R. Schilling, "MaCAN-message authenticated CAN," in *10th Int. Conf. on Embedded Security in Cars (ESCAR 2012)*, 2012.

[64] R. Kurachi, Y. Matsubara, H. Takada, N. Adachi, Y. Miyashita, and S. Horihata, "CaCAN - centralized authentication system in CAN (controller area network)," in *14th Int. Conf. on Embedded Security in Cars (ESCAR 2014)*, 2014.

[65] W. Choi, K. Joo, H. J. Jo, M. C. Park, and D. H. Lee, "VoltageIDS: Low-Level Communication Characteristics for Automotive Intrusion Detection System," *IEEE Transactions on Information Forensics and Security*, 2018.

[66] K.-T. Cho and K. G. Shin, "Fingerprinting Electronic Control Units for Vehicle Intrusion Detection," in *25th USENIX Security Symposium*, 2016.

[67] S. U. Sagong, X. Ying, A. Clark, L. Bushnell, and R. Poovendran, "Cloaking the clock: emulating clock skew in Controller Area Networks," in *Proceedings of the 9th ACM/IEEE International Conference on Cyber-Physical Systems*. IEEE Press, 2018, pp. 32–42.

[68] S. Woo, H. J. Jo, I. S. Kim, and D. H. Lee, "A Practical Security Architecture for In-Vehicle CAN-FD," *IEEE Trans. Intell. Transp. Syst.*, vol. 17, no. 8, pp. 2248–2261, Aug 2016.

[69] A. Mueller and T. Lothspeich, "Plug-and-secure communication for CAN," *CAN Newsletter*, vol. 4, pp. 10–14, 2015.

[70] S. Jain and J. Guajardo, "Physical Layer Group Key Agreement for Automotive Controller Area Networks," in *Conference on Cryptographic Hardware and Embedded Systems*, 2016.

[71] F. Stajano and R. Anderson, "The resurrecting duckling: Security issues for ad-hoc wireless networks," in *International workshop on security protocols*. Springer, 1999, pp. 172–182.

[72] *Specification of Secure Onboard Communication*, R22-11 ed., AUTOSAR, 2022.

[73] S. M. Bellovin and M. Merritt, "Encrypted key exchange: Password-based protocols secure against dictionary attacks," in *Research in Security and Privacy, 1992. Proceedings., 1992 IEEE Computer Society Symposium on*. IEEE, 1992, pp. 72–84.

[74] D. P. Jablon, "Extended password key exchange protocols immune to dictionary attack," in *Proceedings of IEEE 6th Workshop on Enabling Technologies: Infrastructure for Collaborative Enterprises*. IEEE, 1997, pp. 248–255.

[75] C. Miller and C. Valasek, "Adventures in automotive networks and control units," *DEF CON*, vol. 21, pp. 260–264, 2013.

[76] ——, "Remote exploitation of an unaltered passenger vehicle," *Black Hat USA*, vol. 2015, 2015.

[77] G. Bella, P. Biondi, G. Costantino, and I. Matteucci, "TOUCAN: A proTocol tO secUre Controller Area Network," in *Proceedings of the ACM Workshop on Automotive Cybersecurity*. ACM, 2019, pp. 3–8.

[78] C.-W. Lin, Q. Zhu, C. Phung, and A. Sangiovanni-Vincentelli, "Security-aware mapping for CAN-based real-time distributed automotive systems," in *2013 IEEE/ACM International Conference on Computer-Aided Design (ICCAD)*. IEEE, 2013, pp. 115–121.

[79] C.-W. Lin, Q. Zhu, and A. Sangiovanni-Vincentelli, "Security-aware modeling and efficient mapping for CAN-based real-time distributed automotive systems," *IEEE Embedded Systems Letters*, vol. 7, no. 1, pp. 11–14, 2015.

[80] Y. Xie, G. Zeng, R. Kurachi, H. Takada, and G. Xie, "Security/Timing-Aware Design Space Exploration of CAN FD for Automotive Cyber-Physical Systems," *IEEE Transactions on Industrial Informatics*, vol. 15, no. 2, pp. 1094–1104, 2019.

[81] L. Dariz, M. Selvatici, M. Ruggeri, G. Costantino, and F. Martinelli, "Trade-off analysis of safety and security in CAN bus communication," in *2017 5th IEEE International Conference on Models and Technologies for Intelligent Transportation Systems (MT-ITS)*. IEEE, 2017, pp. 226–231.

[82] W. Diffie and M. Hellman, "New directions in cryptography," *IEEE transactions on Information Theory*, vol. 22, no. 6, pp. 644–654, 1976. [Online]. Available: http://doi.org/10.1109/TIT.1976.1055638

[83] H. Wang, D. Forte, M. M. Tehranipoor, and Q. Shi, "Probing attacks on integrated circuits: Challenges and research opportunities," *IEEE Design & Test*, vol. 34, no. 5, pp. 63–71, 2017.

[84] S. Jain, Q. Wang, M. T. Arafin, and J. Guajardo, "Probing Attacks on Physical Layer Key Agreement for Automotive Controller Area Networks (Extended Version)," *arXiv preprint arXiv:1810.07305*, 2018.

[85] Y. Liu, H.-H. Chen, and L. Wang, "Physical layer security for next generation wireless networks: Theories, technologies, and challenges," *IEEE Communications Surveys & Tutorials*, vol. 19, no. 1, pp. 347–376, 2017.

[86] D. Dolev and A. Yao, "On the security of public key protocols," *IEEE Transactions on information theory*, vol. 29, no. 2, pp. 198–208, 1983.

[87] M. Bellare, D. Pointcheval, and P. Rogaway, "Authenticated key exchange secure against dictionary attacks," in *International conference on the theory and applications of cryptographic techniques*. Springer, 2000, pp. 139–155.

[88] F. Hao and S. F. Shahandashti, "The SPEKE protocol revisited," in *International Conference on Research in Security Standardisation*. Springer, 2014, pp. 26–38.

[89] SAE, "High-Speed CAN (HSC) for Vehicle Applications at 500 KBPS," SAE International, Standard, Oct. 2022.

[90] M. Zhang, "Analysis of the SPEKE password-authenticated key exchange protocol," *IEEE Communications Letters*, vol. 8, no. 1, pp. 63–65, 2004.

[91] Q. Tang and C. J. Mitchell, "On the security of some password-based key agreement schemes," in *International Conference on Computational and Information Science*. Springer, 2005, pp. 149–154.

[92] I. P. W. Group *et al.*, "P1363.2: Standard specifications for password-based public-key cryptographic techniques," 2008.

[93] ISO, "ISO/IEC 11770-4. Information technology — Security techniques — Key management — Part 4: Mechanisms based on weak secrets," International Organization for Standardization, Standard, 2nd edition, Nov 2017.

[94] D. G. Steer, L. Strawczynski, W. Diffie, and M. Wiener, "A secure audio teleconference system," in *Proceedings on Advances in Cryptology*, ser. CRYPTO '88. Berlin, Heidelberg: Springer-Verlag, 1990, pp. 520–528. [Online]. Available: http://dl.acm.org/citation.cfm?id=88314.88988

[95] Y. Kim, A. Perrig, and G. Tsudik, "Group key agreement efficient in communication," *IEEE transactions on computers*, vol. 53, no. 7, pp. 905–921, 2004.

[96] X. Ying, G. Bernieri, M. Conti, and R. Poovendran, "TACAN: Transmitter authentication through covert channels in Controller Area Networks," in *Proceedings of the 10th ACM/IEEE International Conference on Cyber-Physical Systems*, 2019, pp. 23–34.

[97] *Requirements on Time Synchronization*, R19-11 ed., AUTOSAR, 2019.

[98] *Specification of Time Synchronization over CAN*, R21-11 ed., AUTOSAR, 2021.

[99] ISO, "ISO11898-4. Road vehicles — Controller Area Network (CAN) — Part 4: Time-triggered communication," International Organization for Standardization, Standard, 1st edition, Aug 2004.

[100] G. Leen and D. Heffernan, "TTCAN: a new time-triggered controller area network," *Microprocessors and Microsystems*, vol. 26, no. 2, pp. 77–94, 2002.

[101] R. I. Davis, A. Burns, R. J. Bril, and J. J. Lukkien, "Controller Area Network (CAN) schedulability analysis: Refuted, revisited and revised," *Real-Time Systems*, vol. 35, no. 3, pp. 239–272, 2007.

[102] V. Berk, A. Giani, G. Cybenko, and N. Hanover, "Detection of covert channel encoding in network packet delays," *Rapport technique TR536, de lUniversité de Dartmouth*, vol. 19, 2005.

[103] Y. Liu, D. Ghosal, F. Armknecht, A.-R. Sadeghi, S. Schulz, and S. Katzenbeisser, "Hide and seek in time—robust covert timing channels," in *European Symposium on Research in Computer Security*. Springer, 2009, pp. 120–135.

[104] S. Cabuk, C. E. Brodley, and C. Shields, "IP covert timing channels: design and detection," in *Proceedings of the 11th ACM conference on Computer and communications security*. ACM, 2004, pp. 178–187.

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

[106] C.-W. Lin, Q. Zhu, and A. Sangiovanni-Vincentelli, "Security-aware mapping for TDMA-based real-time distributed systems," in *Computer-Aided Design (ICCAD), 2014 IEEE/ACM International Conference on*. IEEE, 2014, pp. 24–31.

[107] C.-W. Lin, B. Zheng, Q. Zhu, and A. Sangiovanni-Vincentelli, "Security-aware design methodology and optimization for automotive systems," *ACM Trans. Des. Autom. Electron. Syst.*, vol. 21, no. 1, 2015.

[108] M. R. Moore, R. A. Bridges, F. L. Combs, M. S. Starr, and S. J. Prowell, "Modeling inter-signal arrival times for accurate detection of CAN bus signal injection attacks: a data-driven approach to in-vehicle intrusion detection," in *Proceedings of the 12th Annual Conference on Cyber and Information Security Research*. ACM, 2017, p. 11.

[109] H. M. Song, H. R. Kim, and H. K. Kim, "Intrusion detection system based on the analysis of time intervals of CAN messages for in-vehicle network," in *Information Networking (ICOIN), 2016 International Conference on*. IEEE, 2016, pp. 63–68.

[110] B. H. Bloom, "Space/time trade-offs in hash coding with allowable errors," *Communications of the ACM*, vol. 13, no. 7, pp. 422–426, 1970.

[111] B. Groza and P. Murvay, "Efficient Intrusion Detection With Bloom Filtering in Controller Area Networks," *IEEE Transactions on Information Forensics and Security*, vol. 14, no. 4, pp. 1037–1051, April 2019.

[112] S. B. Moon, P. Skelly, and D. Towsley, "Estimation and removal of clock skew from network delay measurements," in *INFOCOM'99, Proceedings of 18-th Annual Joint Conference of the IEEE Computer and Communications Soc.*, vol. 1. IEEE, 1999, pp. 227–234.

[113] S. Arimoto, "An algorithm for computing the capacity of arbitrary discrete memoryless channels," *IEEE Transactions on Information Theory*, vol. 18, no. 1, pp. 14–20, 1972.

[114] R. Blahut, "Computation of channel capacity and rate-distortion functions," *IEEE transactions on Information Theory*, vol. 18, no. 4, pp. 460–473, 1972.

[115] M. Tian, R. Jiang, C. Xing, H. Qu, Q. Lu, and X. Zhou, "Exploiting temperature-varied ECU fingerprints for source identification in in-vehicle network intrusion detection," in *2019 IEEE 38th International Performance Computing and Communications Conference (IPCCC)*. IEEE, 2019, pp. 1–8.

[116] M. Tian, R. Jiang, H. Qu, Q. Lu, and X. Zhou, "Advanced temperature-varied ECU fingerprints for source identification and intrusion detection in Controller Area Networks," *Security and Communication Networks*, vol. 2020, pp. 1–17, 2020.

[117] S. Kulandaivel, T. Goyal, A. K. Agrawal, and V. Sekar, "CANvas: Fast and Inexpensive Automotive Network Mapping," in *28th {USENIX} Security Symposium ({USENIX} Security 19)*, 2019, pp. 389–405.

[118] G. Baldini, "Voltage Based Electronic Control Unit (ECU) Identification with Convolutional Neural Networks and Walsh–Hadamard Transform," *Electronics*, vol. 12, no. 1, p. 199, 2022.

[119] T. Kohno, A. Broido, and K. C. Claffy, "Remote physical device fingerprinting," *IEEE Transactions on Dependable and Secure Computing*, vol. 2, no. 2, pp. 93–108, 2005.

[120] M. Cristea and B. Groza, "Fingerprinting Smartphones Remotely via ICMP Timestamps," *IEEE Communications Letters*, vol. 17, no. 6, pp. 1081–1083, 2013.

[121] D.-J. Huang, W.-C. Teng, C.-Y. Wang, H.-Y. Huang, and J. M. Hellerstein, "Clock skew based node identification in wireless sensor networks," in *IEEE GLOBECOM 2008-2008 IEEE Global Telecommunications Conference*. IEEE, 2008, pp. 1–5.

[122] C. Arackaparambil, S. Bratus, A. Shubina, and D. Kotz, "On the reliability of wireless fingerprinting using clock skews," in *Proceedings of the third ACM conference on Wireless network security*, 2010, pp. 169–174.

[123] X. Ying, S. U. Sagong, A. Clark, L. Bushnell, and R. Poovendran, "Shape of the Cloak: Formal Analysis of Clock Skew-Based Intrusion Detection System in Controller Area Networks," *IEEE Transactions on Information Forensics and Security*, vol. 14, no. 9, pp. 2300–2314, 2019.

[124] R. M. Gerdes, M. Mina, S. F. Russell, and T. E. Daniels, "Physical-layer identification of wired ethernet devices," *IEEE Transactions on Information Forensics and Security*, vol. 7, no. 4, pp. 1339–1353, 2012.

[125] Q. Xu, R. Zheng, W. Saad, and Z. Han, "Device fingerprinting in wireless networks: Challenges and opportunities," *IEEE Communications Surveys & Tutorials*, vol. 18, no. 1, pp. 94–104, 2015.

[126] M. Foruhandeh, Y. Man, R. Gerdes, M. Li, and T. Chantem, "SIMPLE: Single-frame based physical layer identification for intrusion detection and prevention on in-vehicle networks," in *Proceedings of the 35th Annual Computer Security Applications Conference*, 2019, pp. 229–244.

[127] M. Kneib and C. Huth, "Scission: Signal characteristic-based sender identification and intrusion detection in automotive networks," in *Proceedings of the 2018 ACM SIGSAC Conference on Computer and Communications Security*. ACM, 2018, pp. 787–800.

[128] M. Kneib, O. Schell, and C. Huth, "On the robustness of signal characteristic-based sender identification," *arXiv preprint arXiv:1911.09881*, 2019.

[129] ——, "EASI: Edge-based sender identification on resource-constrained platforms for automotive networks," in *Network and Distributed System Security Symposium (NDSS)*, 2020, pp. 1–16.

[130] M. Kneib and O. Schell, "Effects of the Sampling Technique on Sender Identification Systems for the Controller Area Network," *INFORMATIK 2020*, 2021.

[131] O. Schell and M. Kneib, "VALID: Voltage-Based Lightweight Intrusion Detection for the Controller Area Network," in *2020 IEEE 19th International Conference on Trust, Security and Privacy in Computing and Communications (TrustCom)*. IEEE, 2020, pp. 225–232.

[132] T. Xu, X. Lu, L. Xiao, Y. Tang, and H. Dai, "Voltage Based Authentication for Controller Area Networks with Reinforcement Learning," in *ICC 2019-2019 IEEE International Conference on Communications (ICC)*. IEEE, 2019, pp. 1–5.

[133] M. Rumez, J. Dürrwang, T. Brecht, T. Steinshorn, P. Neugebauer, R. Kriesten, and E. Sax, "CAN Radar: Sensing Physical Devices in CAN Networks based on Time Domain Reflectometry," in *2019 IEEE Vehicular Networking Conference (VNC)*, 2019, pp. 1–8.

[134] J. Zhou, P. Joshi, H. Zeng, and R. Li, "BTMonitor: Bit-time-based Intrusion Detection and Attacker Identification in Controller Area Network," *ACM Transactions on Embedded Computing Systems (TECS)*, vol. 18, no. 6, pp. 1–23, 2019.

[135] J. Ning, J. Wang, J. Liu, and N. Kato, "Attacker identification and intrusion detection for in-vehicle networks," *IEEE Communications Letters*, vol. 23, no. 11, pp. 1927–1930, 2019.

[136] Y. Xun, Y. Zhao, and J. Liu, "VehicleEIDS: A Novel External Intrusion Detection System Based on Vehicle Voltage Signals," *IEEE Internet of Things Journal*, 2021.

[137] L. Xiao, X. Lu, T. Xu, W. Zhuang, and H. Dai, "Reinforcement Learning-Based Physical-Layer Authentication for Controller Area Networks," *IEEE Transactions on Information Forensics and Security*, vol. 16, pp. 2535–2547, 2021.

[138] R. Bhatia, V. Kumar, K. Serag, Z. B. Celik, M. Payer, and D. Xu, "Evading Voltage-Based Intrusion Detection on Automotive CAN," in *NDSS*, 01 2021.

[139] W. Lalouani, Y. Dang, and M. Younis, "Mitigating voltage fingerprint spoofing attacks on the Controller Area Network bus," *Cluster Computing*, pp. 1–14, 2022.

[140] "Dacia Duster Wiring Diagrams," https://www.daciaforum.co.uk/threads/dacia-duster-electrical-wiring-diagrams.39232/, 2021, accessed: 2023-05-22.

[141] "Ford Fiesta Module Communications Network," https://cardiagn.com/2017-2020-ford-fiesta-ewd-module-communications-network/, 2021, accessed: 2023-05-22.

[142] "Ford Kuga Module Communications Network," http://www.fokuman.com/system_diagram-521.html, 2021, accessed: 2023-05-22.

[143] W. Choi, H. J. Jo, S. Woo, J. Y. Chun, J. Park, and D. H. Lee, "Identifying ECUs Using Inimitable Characteristics of Signals in Controller Area Networks," *IEEE Trans. Vehicular Technology*, vol. 67, no. 6, pp. 4757–4770, 2018.

[144] S. Ohira, A. K. Desta, T. Kitagawa, I. Arai, and F. Kazutoshi, "Divider: Delay-Time Based Sender Identification in Automotive Networks," in *2020 IEEE 44th Annual Computers, Software, and Applications Conference (COMPSAC)*. IEEE, 2020, pp. 1490–1497.

[145] S. Ohira, A. K. Desta, I. Arai, and K. Fujikawa, "PLI-TDC: Super Fine Delay-Time Based Physical-Layer Identification with Time-to-Digital Converter for In-Vehicle Networks," in *Proceedings of the 2021 ACM Asia Conference on Computer and Communications Security*, 2021, pp. 176–186.

[146] J. Zhou, G. Xie, S. Yu, and R. Li, "Clock-Based Sender Identification and Attack Detection for Automotive CAN Network," *IEEE Access*, vol. 9, pp. 2665–2679, 2020.

[147] F. Kuhnert, C. Stürmer, and A. Koster, "Five trends transforming the automotive industry," *PricewaterhouseCoopers GmbH Wirtschaftsprüfungsgesellschaft: Berlin, Germany*, vol. 1, no. 1, pp. 1–48, 2018. [Online]. Available: https://www.pwc.com/gx/en/industries/automotive/assets/pwc-five-trends-transforming-the-automotive-industry.pdf

[148] H. K. Fathy, Z. S. Filipi, J. Hagena, and J. L. Stein, "Review of hardware-in-the-loop simulation and its prospects in the automotive area," in *Modeling and simulation for military applications*, vol. 6228. SPIE, 2006, pp. 117–136.

[149] A. Belhocine and M. Bouchetara, "Thermomechanical modelling of dry contacts in automotive disc brake," *International journal of thermal sciences*, vol. 60, pp. 161–170, 2012.

[150] A. Rassõlkin, T. Vaimann, A. Kallaste, and V. Kuts, "Digital Twin for propulsion drive of autonomous electric vehicle," in *2019 IEEE 60th International Scientific Conference on Power and Electrical Engineering of Riga Technical University (RTUCON)*. IEEE, 2019, pp. 1–4.

[151] C. Bosch, "Specification version 2.0," *Published by Robert Bosch GmbH (September 1991)*, 1991.

[152] K. M. Zuberi and K. G. Shin, "Non-Preemptive Scheduling of Messages on Controller Area Network for Real-Time Control Applications," in *Real-Time Technology and Applications Symposium*. IEEE, 1995, pp. 240–249.

[153] M. A. Livani, J. Kaiser, and W. Jia, "Scheduling Hard and Soft Real-Time Communication in a Controller Area Network," *Control Engineering Practice*, vol. 7, no. 12, pp. 1515–1523, 1999.

[154] K. Tindell, A. Burns, and A. Wellings, "Calculating Controller Area Network (CAN) Message Response Times," *IFAC Proceedings Volumes*, vol. 27, no. 15, pp. 35–40, 1994.

[155] M. Farsi, K. Ratcliff, and M. Barbosa, "An introduction to CANopen," *Computing & Control Engineering Journal*, vol. 10, no. 4, pp. 161–168, 1999.

[156] J. Proenza and J. Miro-Julia, "MajorCAN: A Modification to the Controller Area Network Protocol to Achieve Atomic Broadcast." in *ICDCS Workshop on Group Communications and Computations*, 2000, pp. C72–C79.

[157] T. Ziermann, S. Wildermann, and J. Teich, "CAN+: A new backward-compatible Controller Area Network (CAN) protocol with up to 16× higher data rates." in *2009 Design, Automation & Test in Europe Conference & Exhibition*. IEEE, 2009, pp. 1088–1093.

[158] ISO, "ISO11898-2. Road vehicles — Controller Area Network (CAN) — Part 2: High-speed medium access unit," International Organization for Standardization, Standard, 2nd edition, Dec 2016.

[159] F. Hartwich and R. Bosch, "Introducing CAN XL into CAN Networks," *future*, vol. 11898, p. 1, 2015.

[160] M. Grieves, "Completing the Cycle: Using PLM Information in the Sales and Service Functions [Slides]," 2002.

[161] ——, "Digital twin: manufacturing excellence through virtual factory replication," *White paper*, vol. 1, pp. 1–7, 2014.

[162] INCOSE, "Definition for digital twin," 2022. [Online]. Available: https://www.sebokwiki.org/wiki/Digital_Twin_(glossary)

[163] A. M. Madni, C. C. Madni, and S. D. Lucero, "Leveraging Digital Twin Technology in Model-Based Systems Engineering," *Systems*, vol. 7, no. 1, p. 7, 2019.

[164] C. Cimino, E. Negri, and L. Fumagalli, "Review of Digital Twin applications in manufacturing," *Computers in Industry*, vol. 113, p. 103130, 2019.

[165] T. H.-J. Uhlemann, C. Lehmann, and R. Steinhilper, "The Digital Twin: Realizing the Cyber-Physical Production System for Industry 4.0," *Procedia Cirp*, vol. 61, pp. 335–340, 2017.

[166] S. Aheleroff, X. Xu, R. Y. Zhong, and Y. Lu, "Digital Twin as a Service (DTaaS) in Industry 4.0: An Architecture Reference Model," *Advanced Engineering Informatics*, vol. 47, p. 101225, 2021.

[167] A. Hänel, T. Schnellhardt, E. Wenkler, A. Nestler, A. Brosius, C. Corinth, A. Fay, and S. Ihlenfeldt, "The development of a Digital Twin for machining processes for the application in aerospace industry," *Procedia CIRP*, vol. 93, pp. 1399–1404, 2020.

[168] T. R. Wanasinghe, L. Wroblewski, B. K. Petersen, R. G. Gosine, L. A. James, O. De Silva, G. K. Mann, and P. J. Warrian, "Digital Twin for the Oil and Gas Industry: Overview, Research Trends, Opportunities, and Challenges," *IEEE Access*, vol. 8, pp. 104 175–104 197, 2020.

[169] Y. Liu, L. Zhang, Y. Yang, L. Zhou, L. Ren, F. Wang, R. Liu, Z. Pang, and M. J. Deen, "A Novel Cloud-Based Framework for the Elderly Healthcare Services Using Digital Twin," *IEEE Access*, vol. 7, pp. 49 088–49 101, 2019.

[170] R. Magargle, L. Johnson, P. Mandloi, P. Davoudabadi, O. Kesarkar, S. Krishnaswamy, J. Batteh, and A. Pitchaikani, "A Simulation-Based Digital Twin for Model-Driven Health Monitoring and Predictive Maintenance of an Automotive Braking System," in *12th International Modelica Conference, Prague, Czech Republic, May 15-17, 2017*. Linköping University Electronic Press, 2017, pp. 35–46, no. 132.

[171] L. Merkle, A. S. Segura, J. T. Grummel, and M. Lienkamp, "Architecture of a Digital Twin for Enabling Digital Services for Battery Systems," in *2019 IEEE international conference on industrial cyber physical systems (ICPS)*. IEEE, 2019, pp. 155–160.

[172] R. Tharma, R. Winter, M. Eigner *et al.*, "An Approach for the Implementation of the Digital Twin in the Automotive Wiring Harness Field," in *DS 92: Proceedings of the DESIGN 2018 15th International Design Conference*, 2018, pp. 3023–3032.

[173] S. S. Shetty, "Development of a Digital Twin of a Toyota Prius Mk4," Master Thesis, Eindhoven University of Technology, Apr 2022. [Online]. Available: https://pure.tue.nl/ws/portalfiles/portal/202455583/1495305_Shetty.pdf

[174] B. Yu, C. Chen, J. Tang, S. Liu, and J.-L. Gaudiot, "Autonomous Vehicles Digital Twin: A Practical Paradigm for Autonomous Driving System Development," *Computer*, vol. 55, no. 9, pp. 26–34, 2022.

[175] V. Damjanovic-Behrendt, "A Digital Twin-based Privacy Enhancement Mechanism for the Automotive Industry," in *2018 International Conference on Intelligent Systems (IS)*. IEEE, 2018, pp. 272–279.

[176] A. Pokhrel, V. Katta, and R. Colomo-Palacios, "Digital Twin for Cybersecurity Incident Prediction: A Multivocal Literature Review," in *IEEE/ACM 42nd International Conference on Software Engineering Workshops*, 2020, pp. 671–678.

[177] G. P. Sellitto, M. Masi, T. Pavleska, and H. Aranha, "A Cyber Security Digital Twin for Critical Infrastructure Protection: The Intelligent Transport System Use Case," in *IFIP Working Conference on The Practice of Enterprise Modeling*. Springer, 2021, pp. 230–244.

[178] B. Kurt and S. Gören, "Development of a Mobile News Reader Application Compatible with In-Vehicle Infotainment," in *International Conference on Mobile Web and Intelligent Information Systems*. Springer, 2018, pp. 18–29.

[179] D. Lee, K.-S. Kim, and S. Kim, "Controller Design of an Electric Power Steering System," *IEEE Transactions on Control Systems Technology*, vol. 26, no. 2, pp. 748–755, 2017.

[180] E. H. Gurban, B. Groza, and P.-S. Murvay, "Risk assessment and security countermeasures for vehicular instrument clusters," in *2018 48th Annual IEEE/IFIP International Conference on Dependable Systems and Networks Workshops (DSN-W)*. IEEE, 2018, pp. 223–230.

[181] B. Groza, H.-E. Gurban, and P.-S. Murvay, "Designing security for in-vehicle networks: a Body Control Module (BCM) centered viewpoint," in *Dependable Systems and Networks Workshop*. IEEE, 2016, pp. 176–183.

[182] F. Garcia, D. Oswald, T. Kasper, and P. Pavlides, "Lock It and Still Lose It—on the (In)Security of Automotive Remote Keyless Entry Systems," in *25th USENIX Security Symposium*. USENIX Association, 2016, pp. 929–944.

[183] D. F. Oswald, "Wireless Attacks on Automotive Remote Keyless Entry Systems," in *6th International Workshop on Trustworthy Embedded Devices*, 2016, pp. 43–44.

[184] O. A. Ibrahim, A. M. Hussain, G. Oligeri, and R. Di Pietro, "Key is in the Air: Hacking Remote Keyless Entry Systems," in *Security and Safety Interplay of Intelligent Software Systems*. Springer, 2018, pp. 125–132.

[185] L. Wouters, E. Marin, T. Ashur, B. Gierlichs, and B. Preneel, "Fast, Furious and Insecure: Passive Keyless Entry and Start Systems in Modern Supercars," *IACR Transactions on Cryptographic Hardware and Embedded Systems*, vol. 2019, no. 3, pp. 66–85, 2019.

[186] J. Dürrwang, J. Braun, M. Rumez, and R. Kriesten, "Security Evaluation of an Airbag-ECU by Reusing Threat Modeling Artefacts," in *2017 International Conference on Computational Science and Computational Intelligence (CSCI)*. IEEE, 2017, pp. 37–43.

[187] H. Mun, K. Han, and D. H. Lee, "Ensuring Safety and Security in CAN-Based Automotive Embedded Systems: A Combination of Design Optimization and Secure Communication," *IEEE Transactions on Vehicular Technology*, vol. 69, no. 7, pp. 7078–7091, 2020.

[188] M. K. Ishak and F. K. Khan, "Unique Message Authentication Security Approach based Controller Area Network (CAN) for Anti-lock Braking System (ABS) in Vehicle Network," *Procedia Computer Science*, vol. 160, pp. 93–100, 2019.

[189] L. Guo, J. Ye, and L. Du, "Cyber–Physical Security of Energy-Efficient Powertrain System in Hybrid Electric Vehicles Against Sophisticated Cyberattacks," *IEEE Transactions on Transportation Electrification*, vol. 7, no. 2, pp. 636–648, 2020.

[190] J. Ye, L. Guo, B. Yang, F. Li, L. Du, L. Guan, and W. Song, "Cyber–Physical Security of Powertrain Systems in Modern Electric Vehicles: Vulnerabilities, Challenges, and Future Visions," *IEEE Journal of Emerging and Selected Topics in Power Electronics*, vol. 9, no. 4, pp. 4639–4657, 2020.

[191] U. Hoff and D. Scott, "Challenges for wiring harness development," *CAN Newsletter*, pp. 14–19, 2020.

[192] A. Zhou, Z. Li, and Y. Shen, "Anomaly Detection of CAN Bus Messages Using a Deep Neural Network for Autonomous Vehicles," *Applied Sciences*, vol. 9, no. 15, p. 3174, 2019.

[193] M. D. Hossain, H. Inoue, H. Ochiai, D. Fall, and Y. Kadobayashi, "LSTM-Based Intrusion Detection System for In-Vehicle Can Bus Communications," *IEEE Access*, vol. 8, pp. 185 489–185 502, 2020.

[194] T. Andreica, C.-D. Curiac, C. Jichici, and B. Groza, "Android Head Units vs. In-Vehicle ECUs: Performance Assessment for Deploying In-Vehicle Intrusion Detection Systems for the CAN Bus," *IEEE Access*, vol. 10, pp. 95 161–95 178, 2022.

[195] H. Kim, Y. Jeong, W. Choi, D. H. Lee, and H. J. Jo, "Efficient ECU Analysis Technology Through Structure-Aware CAN Fuzzing," *IEEE Access*, vol. 10, pp. 23 259–23 271, 2022.

[196] T. H. Aldhyani and H. Alkahtani, "Attacks to Automatous Vehicles: A Deep Learning Algorithm for Cybersecurity," *Sensors*, vol. 22, no. 1, p. 360, 2022.

[197] R. Islam, M. K. Devnath, M. D. Samad, and S. M. J. Al Kadry, "GGNB: Graph-based Gaussian naive Bayes intrusion detection system for CAN bus," *Vehicular Communications*, vol. 33, p. 100442, 2022.

[198] T. Toyama, T. Yoshida, H. Oguma, and T. Matsumoto, "PASTA: Portable Automotive Security Testbed with Adaptability," *London, blackhat Europe*, 2018.

[199] C. Gay, T. Toyama, and H. Oguma, "Resistant Automotive Miniature Network," in *Chaos Computer Congress*, 2020.

[200] S. Hariharan, A. V. Papadopoulos, and T. Nolte, "On in-vehicle network security testing methodologies in construction machinery," in *27th International Conference on Factory Automation (ETFA)*, 2022, pp. 1–4.

[201] T. Brennich and M. Moser, "Putting Automotive Security to the Test," *ATZelectronics worldwide*, vol. 15, no. 1, pp. 46–51, 2020.

[202] P. Joshi, S. Ravi, Q. Liu, U. D. Bordoloi, S. Samii, S. K. Shukla, and H. Zeng, "Approaches for assigning offsets to signals for improving frame packing in CAN-FD," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 39, no. 5, pp. 1109–1122, 2019.

[203] H. Lim, G. Kim, S. Kim, and D. Kim, "Quantitative analysis of ringing in a Controller Area Network with Flexible Data rate for reliable physical layer designs," *IEEE Transactions on Vehicular Technology*, vol. 68, no. 9, pp. 8906–8915, 2019.

[204] S. Kang, J. Seong, and M. Lee, "Controller Area Network with Flexible Data rate transmitter design with low electromagnetic emission," *IEEE Transactions on Vehicular Technology*, vol. 67, no. 8, pp. 7290–7298, 2018.

[205] R. Shirai, K. Wada, and T. Shimizu, "CAN signal processing for EMI reduction cooperating with switched-mode power supply," in *2020 IEEE Energy Conversion Congress and Exposition (ECCE)*. IEEE, 2020, pp. 3566–3571.