fhwedel
UNIVERSITY OF APPLIED SCIENCES

Seminar IT-Security

WS 2016/2017

CAN-Bus Security

submitted by

Thomas Robert Döbbert

(ite101932@fh-wedel.de)

University of Applied Science Wedel

Master IT-Engineering

Advisor: Prof. Dr. Gerd Beuster (gb@fh-wedel.de)

Hamburg, March 3rd 2017

fhwedel
UNIVERSITY OF APPLIED SCIENCES

Contents

1 Introduction

CAN (Controller Area Network) is a serial communication technology used especially for reliable data exchange between electronic control units (ECUs) in the automobile. An impressive example is the Mercedes S-Class with 170 ECUs.[3]
The CAN bus is characterized as a broadcast type bus, which means that all nodes can “hear“ all transmissions. While sending a message to just a specific node, all other nodes will listen to the traffic. CAN hardware offers local filtering, though, each node may react only on the interesting messages. The bus uses Non-Return To Zero (NRZ) with bit-stuffing. While NRZ describes in CAN that only positive and negative voltage levels are used, bit-stuffing describes the insertion of non information bits. The modules are wired to the bus and configured in the specific way that if one node is driving the bus to a logical 0, the whole bus is in that state regardless of nodes transmitting a logical 1.
For error handling, CAN standard defines an elaborate scheme and confinement, described in section 2.3.2. Bit timing and clock synchronization are essential for CAN bus configuration, but will not be discussed in detail. Different physical layers can be used to implement CAN, which is described in the CAN Standard discussed in Section 1.2 [4].

1.1 History

Robert Bosch GmbH started developing CAN in 1983. Officially, the CAN protocol was released in 1986 at the Society of Automotive Engineers (SAE) conference in Detroit, Michigan. In 1987, the first CAN controller chips, produced by Intel and Philips, came to the market. The first vehicle using the CAN-based multiplex wiring was the BMW 8 Series. In 1991, Bosch published the latest version of the CAN specification. The specification has two parts. Part A holds the standard format with an 11-bit identifier, called CAN 2.0A, and part B using an extended CAN device with 29-bit identifiers, called CAN 2.0B.[2]

1.2 Standardization

In the early 1990s, the Bosch CAN specification (version 2.0) was submitted for international standardization. In November 1993, the ISO 11898 standard was published after several political discussions concerning the Vehicle Area Network (VAN). The VAN was a communication protocol developed by the French car manufacturers. Next to the CAN protocol also a physical layer for bit-rates up to 1 Mbit/s was standardized. Additionally, a low-power, fault-tolerant way of data transmission using CAN was standardized in ISO 11519-2. The extended frame format using 29-bit CAN identifier was standardized in 11898-2 in 1995.
Since all published CAN specifications and standardizations contained errors or were incomplete, Bosch made sure (and still does) that all CAN chips satisfy the Bosch CAN reference model. Therefore, used test patterns based on the internationally standardized test specification ISO 16845 were created. There are several test houses offering CAN conformance testing services.
Revised CAN specifications have been ISO 11898-1 describing the CAN data link layer, ISO 11898-2 defining the Non-fault-tolerant CAN physical layer, and ISO 11898-3 specifying the Fault-tolerant CAN physical layer. In addition, CAN-based application profiles based on the US-protocol J1939, which are not compatible, have been specified in ISO standard 11992 (truck and trailer interface) and 11783 (agriculture and forestry machines).[5]

2 Basics

In comparison to USB or Ethernet, CAN does not send large blocks of data point-to-point from node A to node B under the supervision of a central bus master. In a CAN network, many short messages are broadcast to the entire network. This provides data consistency in every node of the system.

2.1 The CAN Standard

CAN defines serial communication to replace the complex wiring harness with a two-wire bus. The CAN communication protocol, ISO-11898 from 2003 specifies how information is passed between devices on a network and matches to the Open Systems Interconnection (OSI) model, which is defined in terms of layers. The physical layer describes the actual communication between the devices connected to the physical medium. The bottom two layers of the seven layer OSI/ISO model are defined as data-link layer and the physical layer in the ISO 11898 architecture. This is shown in Figure 2.1 below:

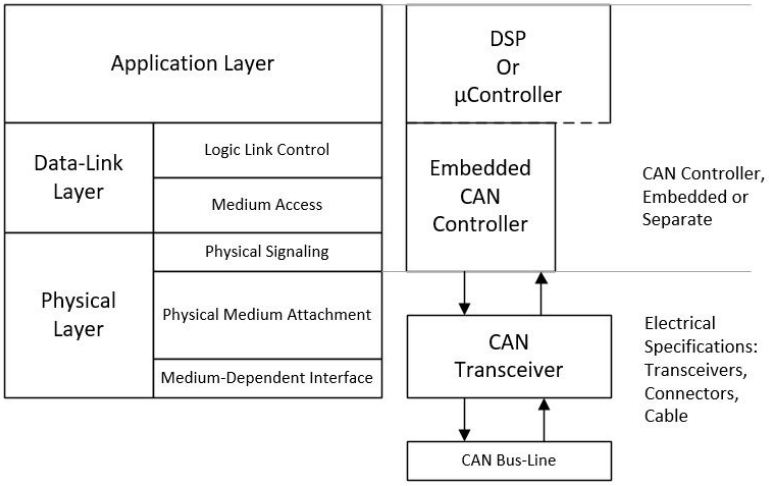


Figure 2.1: Layered ISO standard architecture redrawn[6]

In Figure 2.1, the application layer establishes the communication link to an upper-level application specific protocol such as the vendor-independent CANopen or DeviceNet.[6]

2.2 Standard CAN or Extended CAN

The CAN communication protocol is a carrier-sense, multiple access protocol with collision detection and arbitration on message priority (CSMA/CD+AMP), where each node on a bus must wait for a prescribed period of inactivity before attempting to send a message(CSMA) and collisions are resolved through a bit-wise arbitration, based on a pre-programmed priority of each message in the identifier field of a message (CD+AMP). The higher priority identifier always wins bus access. The last logic-high in the identifier keeps on transmitting. An arbitrating node knows if it is placed as the logic-high bit on the bus, since every node on a bus takes part in writing every bit.
The standard 11-bit identifier is stated in the ISO-11898:2013 Standard providing data rates from 125 kbps to 1 Mbps and for 2^{11} or 2048 different message identifiers. The later improved "extended" 29-bit identifier provides for 2^{29} or 537 million identifiers message fields.[6]

2.2.1 Standard CAN

The Standard CAN 11-Bit Identifier can be seen in the Figure 2.2 below:

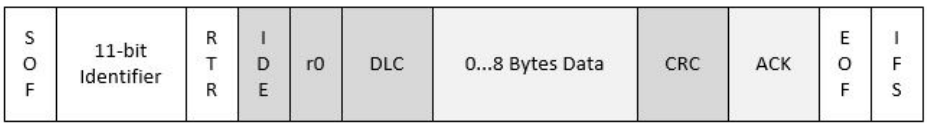


Figure 2.2: Standard CAN 11-Bit Identifier redrawn[6]

The single dominant start of frame (SOF) bit marks the start of a message and synchronizes the nodes on a bus after being idle. The 11-bit identifier sets the priority of the message; the lower the binary value, the higher its priority. The single remote transmission request (RTR) bit is dominant if information is required from another node. All nodes receive the request, whereas the identifier determines the specific node. At all time, all data in the system is uniform. If the identifier extension (IDE) is dominant, a standard CAN identifier with no extension is being transmitted. r0 is a reserved bit for future use. The 4-bit data length code (DLC) contains the number of bytes of data being transmitted. Next the actual data up to

64 bits can be transmitted. The 16-bit (15 bits plus delimiter) cyclic redundancy check (CRC) holds the checksum of the application data for error detection. The acknowledgment bit is used for receiving the message and overwriting this bit to indicate an error-free message has been sent. If an error has been discovered, this bit is left recessive, it discards the message and the sending node repeats the message after rearbitration. The integrity of the data is therefore established. The second bit of the ACK is for a delimiter. The 7-bit end-of-frame(EOF) field marks the end of a CAN frame message and disables bit-stuffing or indicating a stuffing error if dominant. The 7-bit interframe space(IFS) field includes the required time by the controller to move received frames to its position in a message buffer area.[6]

2.2.2 Extended CAN

The Extended CAN message can be seen in Figure 2.3 below:

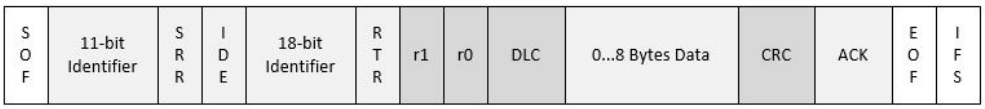


Figure 2.3: Extended CAN 29-Bit Identifier redrawn[6]

The Extended CAN message is similar to the Standard CAN message except of: The RTR is replaced by the substitute remote request(SRR) bit as a placeholder. A recessive bit in the identifier extension(IDE) indicates that more identifier bits follow (18-bit identifier). An additional reserve (r1) has been added.[6]

2.3 A CAN Message

2.3.1 Arbitration

A fundamental characteristic of CAN is that it uses opposite logic states for being dominant and recessive in comparison to the general approach. If the bit is zero its dominant and if the bit is one its recessive. This is illustrated in Figure 2.4. Therefore, transceiver have the driver input and receiver output pins passively pulled high internally. If no input is present, the device automatically defaults to a recessive bus state on all input and output pins.

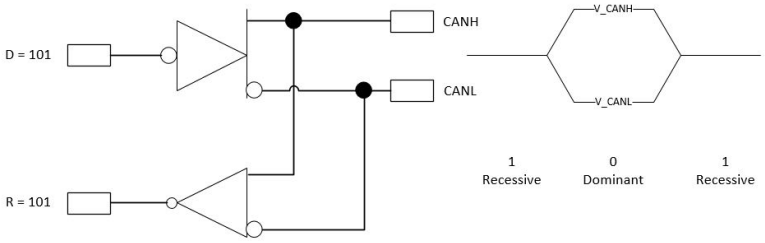


Figure 2.4: The Inverted Logic of a CAN Bus redrawn[6]

In a CAN network, the bus access is event-driven and randomly. The decision of who gets access to the bus if two nodes are trying simultaneously is done nondestructive and with bit-wise arbitration. Nondestructive means that the node wining the arbitration just continuous with the message transfer. The real-time feature is given through the allocation of priority to messages in the identifier, whereas the higher the binary message identifier number, the lower its priority. A CAN controller automatically handles the arbitration process, which is illustrated in Figure 2.5 below:

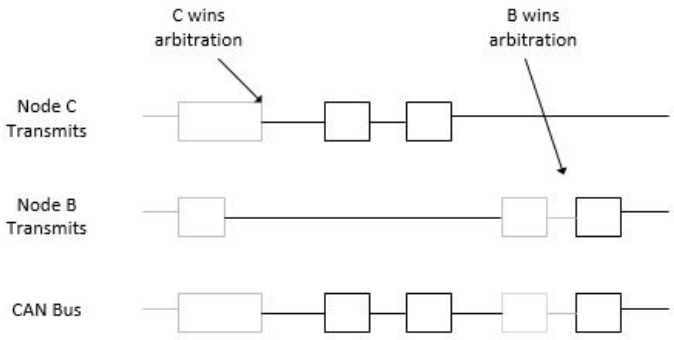


Figure 2.5: Arbitration on a CAN Bus redrawn[6]

All nodes continuously monitor the bus and therefore node B stops transmission, because it detects that node C has a higher priority dominant bit. Node B detects that the bus state does not match the bit that is transmitted and waits until node C has finished its transmission. This functionality is stated in ISO 11898 physical

signaling layer, which means that it is transparent to all CAN users.[6]

2.3.2 Frames

There are four different frames that can be transmitted on CAN bus, which are the data frame, the remote frame, the error frame, and the overload frame.

Data Frame

The data frame is the most common message type and includes the Arbitration Field, the Data Field, the CRC Field, and the Acknowledgment Field. The Arbitration Field can contain an 11-bit(2.2) or 29-bit(2.3) identifier and the RTR bit, which is dominant for data frames. The Data Field contains zero to eight bytes of data and the CRC Field includes the 16-bit checksum for error detection. Finally, the Acknowledgment Field (ACK) builds the end of the frame.

Remote Frame

The remote frame is used to offer the transmission of data from another node. There are only two differences to the data frame. The first difference is that the RTR bit is set to recessive in the arbitration field and the second difference is that there is no data.

Error Frame

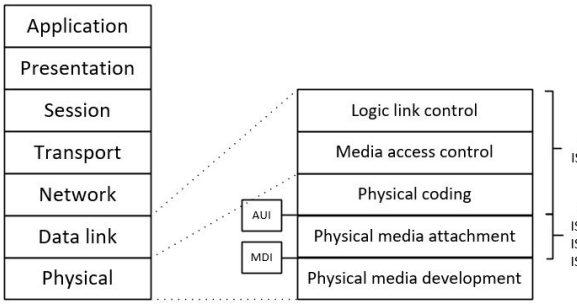
The error frame violates the formatting rules of a CAN message because its a special message. An error message will be transmitted, if there was an error detected and it causes all other nodes in the network to send also an error frame. The original transmitter then automatically retransmits the message. To ensure that a node cannot occupy a bus by repeatedly sending error frames, an error counter is used.

Overload Frame

The overload frame is similar to the error frame and is transmitted by a node that becomes too busy. It is often used to create an extra delay between messages.[6]

2.4 Architecture

The existing OSI reference model is used by CAN to transfer data among nodes that are connected in a network. The OSI model represents seven layers through which the data is passed during communication between devices in a network. As can be seen in Figure 2.6, CAN uses the lower two layers of the OSI model; physical layer and data link layer. The other five layers are left out by BOSCH CAN specification for individual optimization and adaption according to the needs of system designer.



Pin	Signal	Description
1	–	Reserved for CiA
2	CAN_L	CAN bus-line dominant low
3	CAN_GND	CAN ground
4	–	Reserved for CiA
5	(CAN_SHLD)	Optional CAN shield
6	(GND)	Optional power ground
7	CAN_H	CAN bus-line dominant high
8	–	Reserved for CiA
9	(CAN_V+)	Optional (external) power supply

Table 2.1: 9-pin D-Sub connection redrawn[5]

Figure 2.6: CAN Layer Presentation redrawn[5]

As shown in Figure 2.6 above, the physical coding (PCS) implemented in the CAN controller chips, the physical media attachment (PMA) specifying the transceiver characteristics, and the physical media-dependent sub-layer (PMS) form the CAN physical layer. The PMS is application-specific and generally not standardized. The bit-encoding and decoding, the synchronization, and the bit timing is comprised by the PCS. The interface to the transceiver chips is provided by this sub-layer called the attachment unit interface (AUI). The medium-dependent interface (MDI) is the interface to the physical bus-lines, which is generally the well-known 9-pin D-sub connector (DIN 41652) shown in Table 2.1 [5].

3 Applications

3.1 CAN in Automation CiA

The main field of application is the Automotive area. A modern car has as many as 70 or more electronic control units (ECUs) for various subsystems. The biggest processor is in general the engine control unit. The different ECUs are demonstrated in Figure 3.1 below:

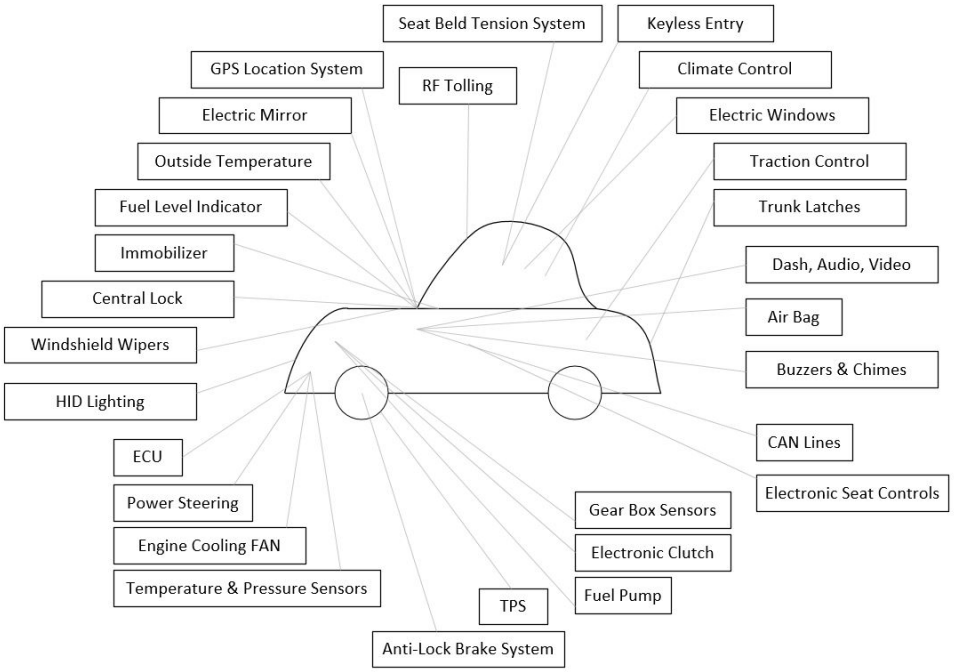


Figure 3.1: CAN in Automotive redrawn[14]

The interconnection between different vehicle systems allows a wide range of safety, economy and convenience features to be implemented using only software. "Hard wiring" these features for more functionality would add cost and complexity as well as weight to the car. Some example are Auto Start/Stop, electric park brakes, parking assist systems, auto lane assist/collision avoidance systems or auto brake wiping.
Additionally, it should be mentioned that LIN bus standard has been implemented for non-critical subsystems such as air-conditioning and infotainment. In these

applications data transmission speed and reliability are less critical.[2]

3.2 Other

The CAN bus protocol is also used as a fieldbus in general automation environments such as i.e. Rofin Lasers 3.2 due to the low cost of CAN controllers and processors. Another application is the Shimano Di2 electronic gear shift system 3.4 for road bicycles since 2009. Also NISMO aims to use CAN bus to create real-life racing laps in the videogame Gran Turismo 6 3.3 using the game's GPS Data Logger function. This would allow players to race against real laps.[2]

Figure 3.2: Rofin Laser[16] **Figure 3.3:** Gran Turismo[17] **Figure 3.4:** Shimano Di2[18]

4 Security

The security of CAN was based on the security of data transfer which is handled by the error detection, error signaling and self-monitoring in the protocol. As CAN bus is a broadcast type bus, which means when accessing one component with access to the CAN bus, allows the entire vehicle network to be compromised. Figure 4.1 shows recorded data from the physical CAN bus using an oscilloscope to measure the differential voltage across the bus. The different elements of the CAN frame are highlighted to show how easy it is to read this data. The first bits after the start-of-frame bit five the CAN ID field (**0x123**) and the bits for the data payload can be easily interpreted to produce the message **0x DE-AD-BE-EF**. Once physical access to the bus has been established data can be read or sent.

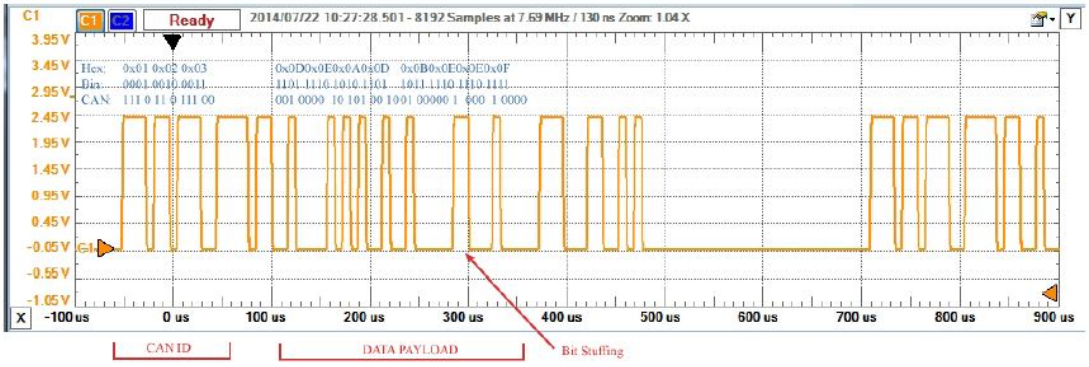


Figure 4.1: Easy Data Access on the CAN Bus

4.1 Attacks

There are different types of attacks such as:

- **Eavesdropping** means that unauthorized nodes read data.
- **Modification** is done if authorized nodes change data or software.
- **Spoofing** is done if unauthorized nodes send authorized messages or reply messages.

4.2 CAN Security Concerns and Challenges

Some of the very features that are extolled as advantage of CAN, make security aspects of CAN very challenging([7] [8] [9] [10] [11]):

- Since all messages are part of a broadcast transmission, all nodes receive these messages and then filter them for relevance meaning. This make eavesdropping very easy by inserting a rogue ECU or simply manipulating an existing ECU.
- There is no authentication mechanism because there is no way to identify the message origins. The message identifier does not give an indication of the sender, that way the attacker can send fake messages with high priority to compromise the bus or to cause problems by sending fake error frames.
- Most systems employ a single gateway to support interconnection of different buses and local gateways between homogeneous networks. Therefore, the entire network can be threaten when a single component on any bus would be compromised.
- Strong security primitives require large payloads. There are limits since the standard CAN frame is 8 bytes. Some researchers claim that CAN bus might be unsuitable for secure communication and that Ethernet with IPsec may offer a better solution for security concerns.
- A denial-of-service (DoS) attack can easily be achieved since a node can put the bus in a dominant state which will prevent any other node from sending messages due to the arbitration scheme.
- Limitations are set by the hard real-time constraints of many CAN bus environments.
- Non-repudiation problem is not solved up to now, which means it cannot be proved that a particular ECU did not send or receive a particular message.
- CAN bus is so widely used that lots of different hardware and software is available for cheap equipment prices to access the on-board diagnostics (OBD) protocol.

Other security issues arise when security standards and regulations are ignored ([8] [9]):

- Reflashing an ECU whilst operational.
- The same reflash keys are used for more than one ECU.
- Reflashing a CAN gateway from a low speed network when this is prohibited.
- Easy retrieval of keys from ECUs.

4.3 Desired Properties and Countermeasures

When talking about security, the model of CIA (confidentiality, integrity, availability) defines the overall security objectives: The three most desired properties for

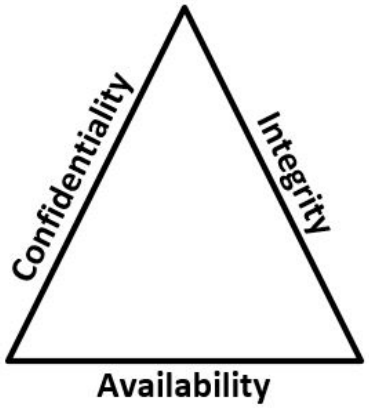


Figure 4.2: The Information Security triad: CIA.

security are described in the Figure 4.2 above as:

- **Confidentiality** data is not read by unauthorized nodes
- **Data Integrity** data is not changed by unauthorized nodes
- **Availability** sender and receiver verify each other‘s identity

The overall security objectives describe also for the automotive context the confidentiality and integrity of data, integrity of hardware and software, availability of data and

services, and uniqueness of hardware components (no cloning). In the context of vehicle-to-infrastructure (V2I) security solutions, additionally, authentication and freshness are important security features.[12]
When examining message authentication over CAN, Hartkopp, Reuber, and Schilling (2012)[13] identify four requirements for CAN bus security: message authentication and freshness, real-time capabilities, flexibility to adjust security for different message types, and backward compatibility so that nodes without security will still work after authentication is introduced. They argue that data integrity and authentication is arguably more important than confidentiality, because, in automotive applications, it may be more important to maintain proper control of the vehicle than protecting data.
Countermeasures against security issues regarding the CIA model are:

- **Confidentiality:** Cryptography
- **Data Integrity:** Message Authentication Codes (MACs)
- **Authentication:** Authentication nodes and messages by identity

4.4 Secure CAN Communication

Out of the previous section can be derived that the following security concepts are needed [15]:

- **Firewall** for policy-based filtering
- **IDS** for anomaly detection
- **Cryptography** for authentication & encryption

The presentation of Timo van Roermund at the ESCAR in November 2016 citeNXP2016 showed that a combination of these security concepts are planned for future use in industry.

4.4.1 Network Layer Firewall

Packets can be filtered through their frames by three different approaches. One is the stateless or intra-message approach, where the CAN ID is checked. Therefore,

a white- or/and black list can be used as illustrated in Figure 4.3 below:

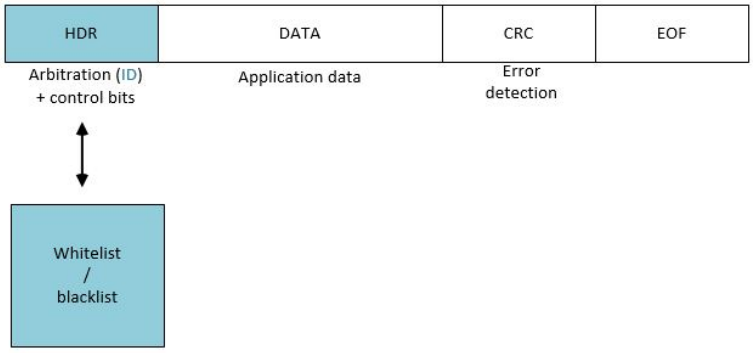


Figure 4.3: Packet(Frame) Filtering redrawn[15]

Another approach is the stateful or inter-message inspection, but this approach is not applicable to CAN since in CAN are no sessions.
In general all incoming and/or outgoing frames can be checked.
As extension to the filtering, the frame rates should be limited, which prevents too many frames from being send or received. This is also a countermeasure against denial-of-service attacks.[15]

4.4.2 Network Intrusion Detection Systems (NIDS)

The network traffic needs to be inspected regarding suspicious data and patterns. Suspicious data can be inspected by intra-message packet inspection, whereas suspicious patterns are checked through inter-message packet inspection. There are two types of NIDS. One type is signature-based, which checks against a database for known malicious patterns. Another type is anomaly-based, which is verifying against a model of trustworthy traffic and is created using machine learning.
Since it is risky to actively block messages, which is done in Network Intrusion Protection (NIPS), messages will only be tagged. Actively blocking messages may suffer from false negatives (malicious data going detected), as well as from false positives (trustworthy data tagged as suspicious) [15].

4.4.3 Cryptographic Protection

The Standard CAN frames as mentioned before only include protection against unintentional errors like bit flips. It is important that the message is authenticated regarding origin and integrity. Therefore, additional data must be exchanged in form of a MAC plus some token against replay attacks. This is complicated in the short Standard CAN format. The application data would lose some bytes to the MAC field. The frame can be seen in Figure 4.4 below: It might be possible to

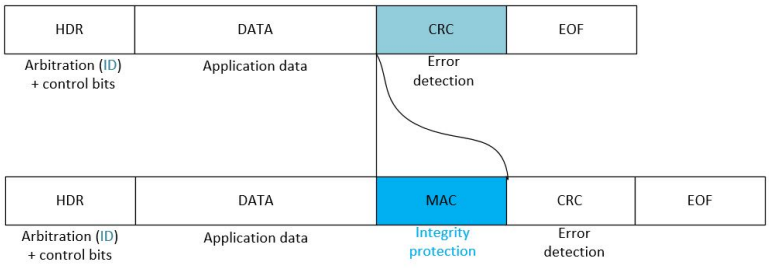


Figure 4.4: Frame for Message Authenication redrawn[15]

implement security on a higher protocol layer, but this might result in real-time issues.
The message confidentiality needs to be protected by using message encryption:

Figure 4.5: Frame for Message Encryption redrawn[15]

There are three main approaches to deliver authenticated encryption:

- **MAC-then-Encrypt:** This is done in TLS where the MAC is produced over plaintext and then it is encrypted with the plaintext.
- **Encrypt-then-MAC:** This is used in IPsec where the plaintext is encrypted, the MAC is computed on the cipthertext, and appended to the ciphertext. Also the initialization vector (IV) and the encryption method identifier is included into the MACed data.

- **Encrypt-and-MAC:** The plaintext is encrypted and the MAC is computed over the original plaintext.

Each of these has its potential security weaknesses.
MAC-then-Encrypt:

- No integrity on the ciphertext, since there is no knowledge until the message is decrypted whether the message was authentic or spoofed.
- Plaintext integrity
- Theoretically, if the cipher scheme is malleable it may be possible to alter the message to appear valid and have a valid MAC code
- MAC cannot provide any information on the plaintext because it is encrypted.

Encrypt-then-MAC:

- Provides integrity of ciphertext.
- Plaintext integrity.
- In case that the cipher scheme is malleable, there are no concerns, since the MAC code will filter out this invalid ciphertext.
- The MAC does not provide any information on the plaintext.

Encrypt-and-MAC:

- No integrity on the ciphertext, since there is no knowledge until the message is decrypted whether the message was authentic or spoofed.
- Plaintext integrity.
- Theoretically, if the cipher scheme is malleable it may be possible to alter the message to appear valid and have a valid MAC code.
- May reveal information about the plaintext in the MAC.

Encrypt-then-MAC is the most ideal scenario even tough Fergusion, Schneider and Kohno[1] argue that MAC-then-encrypt is the “natural“ order and that encrypt-then-MAC is rather complex. In Encrypt-then-MAC before decryption, any modification

to the ciphertext without valid MAC code can be filtered out. This protects against any attacks on the implementation as well as the MAC cannot be used to get knowledge about the plaintext[8]. There is an “almost“ consensus on using encrypt-then-MAC.

4.4.4 CAN Security Concepts Comparison

A comparison of the different security concepts can be seen in Figure ?? below:

	Network Layer Firewall	**NIDS**	**Cryptographic Protection**
Confidentiality (e.g. eavesdropping)	No	No	Yes
Integrity (e.g. spoofing, MITM)	Most common threats	Most threats	Most threats
Availability (e.g. flooding/DoS)	Yes	Yes (NIPS)	Most threats
Implementation challenges	Secure (re-) configuration	Secure (re-)configuration, False positives & false negatives	Data overhead, performance, key management, lack of standards (interoperability)
Apparent Value	Detect and block attacks – **When** they happen – in the **Vehicle**	Early detection attacks – **Before** they happen – in the **Cloud**	Detect and block attacks – **When** they happen – in the **Vehicle**

Table 4.1: Comparison of CAN Security Concepts redrawn[15]

The concepts are rather complementary and serve therefore different needs. In practice, a combination of these concepts needs to be accomplished[15].

4.4.5 Example: Protect against Spoofing Attacks

Figure 4.6 shows a legitimate node A which sends a message with CAN ID 0x123 and hostile node E sends also a message with the same CAN ID, but different payload data. Node B receives both messages, but cannot see their origin. Therefore, node B will process both messages.

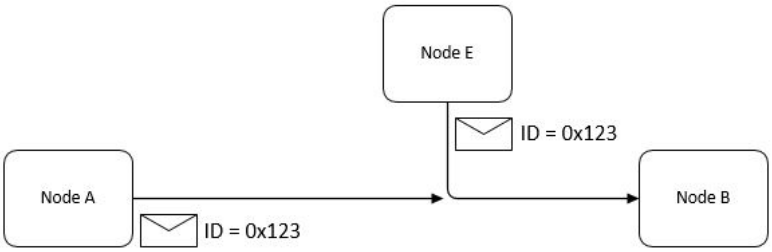


Figure 4.6: Message Sending with same CAN ID redrawn[15]

First solution would imply that sender blocks the outgoing message by using a firewall and/or intrusion detection system (IDS). In the second solution, the receiver authenticates the incoming message and rejects because of a wrong MAC. In the third solution the genuine sender detects spoofing and blocks the message. These solutions can be seen in Figure 4.7, 4.8 and 4.9 below:

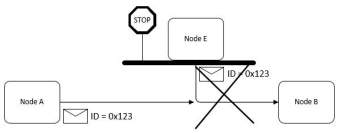


Figure 4.7: Solution 1: Sender blocks outgoing message redrawn[15]

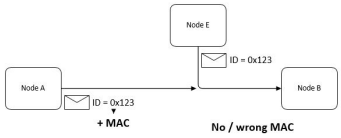


Figure 4.8: Solution 2: Receiver authenticates incoming message redrawn[15]

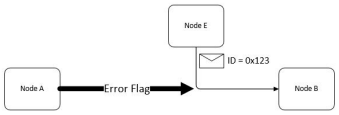


Figure 4.9: Solution 3: genuine sender detects spoofing and blocks message redrawn[15]

4.4.6 Example: Protect against Denial-of-Service Attacks

In CAN bus protocol, high priority identifiers must be used to win arbitration. But if the bus is overloaded, messages cannot be handled any longer. Therefore, a priority-dependent rate limitation is needed. The weighted moving average can be calculated and used block overload by increasing on transmission. The CAN ID can be taken as reference and decreased over time. This attack is illustrated in Figure 4.10 below:

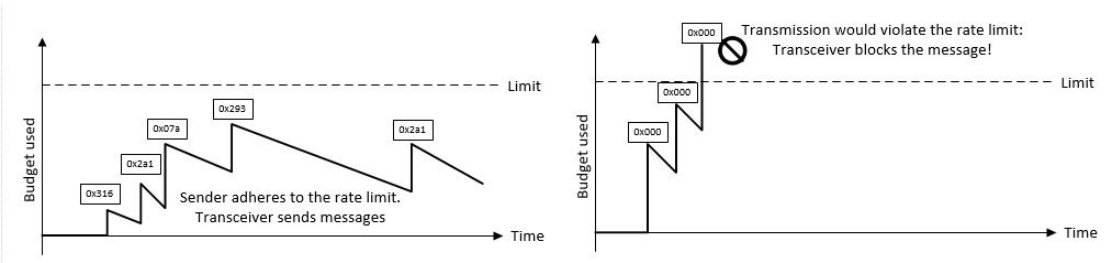


Figure 4.10: Denial-of-Service Attack redrawn[15]

4.4.7 Implementation

The question is where to implement the security regarding authentication, filtering, and encryption. One option is to design a new ECU with security support regarding

a Secure Hardware Extension (SHE) or a Hardware Security Module (HSM). But in this case, each vendor has to redesign their microcontroller unit. An easier approach would be to upgrade the existing ECUs and implement the security into the CAN transceiver. Another advantage is that the security would still be intact, even though the host is compromised [15]. The solutions are shown in Figure 4.11 below:

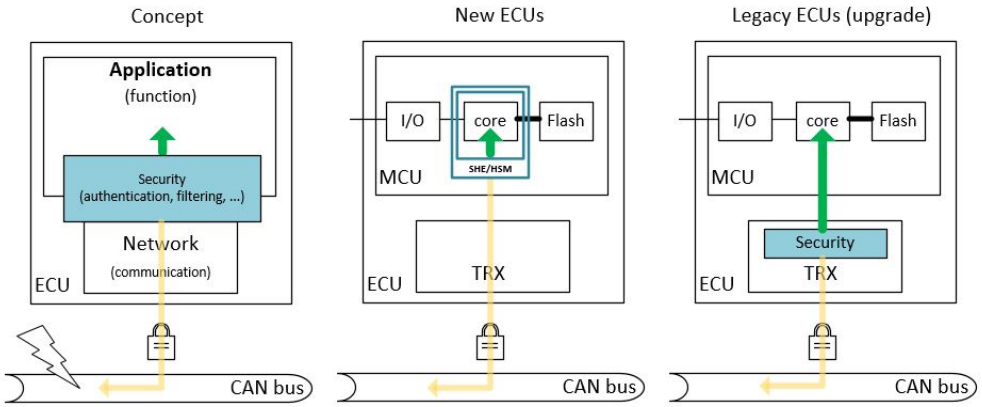


Figure 4.11: Implemention of CAN Security Hardware redrawn[15]

5 Outlook

As this seminar work showed, there is an immense need for CAN security. But there are big boundaries to overcome. Lot's of different industries are involved into the process of changing and standardizing a secure solution. Another issue is the real-time requirement and the short message length of CAN bus protocol, which makes secure solutions complicated. It seems like that the short-term need to secure CAN bus protocol could be to use CAN FD (flexible data rate). CAN FD bus could give a better solution to implement authentication and encryption methods. The application data would still be acceptable long. But the mid to long-term need would be to use Secure Ethernet to be more flexible in data length and to use sessions. Furthermore, various protocol stacks and various security solutions, such as IPsec, could be used [15].
With the invention of CAN FD protocol, the lifetime of CAN might have been prolonged by 10 to 20 years. Next generation of in-vehicle networks are already planned to use CAN FD protocol. The advantage of configurable payload length from 0 to 64 byte makes CAN FD flexible.
In addition CANopen FD protocol for CAN FD lower-layer is in development. This is used especially for industrial motion control application, higher transmission rates and longer payloads. The development of a CAN FD based application layer for commercial vehicle using the existing Parameter Groups (SAE J1939) is also in development.[5]

Bibliography

[1] Ferguson, Schneider, Kohno (March 2010).
Cryptography Engineering. Design Principles and Practical Applications.

[2] Zimmermann, W., & Schmidgall, R. (2014).
Bussysteme in der Fahrzeugtechnik. *Protokolle, Standards und Softwarearchitektur* (5th Edition).

[3] Vector Informatik GmbH
URL: https://elearning.vector.com/vl_can_introduction_en.html

[4] Kvaser, CAN Protocol Tutorial
URL: https://www.kvaser.com/can-protocol-tutorial/

[5] CAN Cia, History of CAN technology
URL: https://www.can-cia.org/can-knowledge/can/can-history/

[6] Texas Instruments, Introduction to the Controller Area Network (CAN)
URL: http://www.ti.com/lit/an/sloa101a/sloa101a.pdf

[7] Groll, A., & Ruland, C. (2009).
Secure and authentic communication on existing in-vehicle networks. In *Intelligent vehicles symposium, 2009 IEEE* (p. 1093–1097).

[8] Hazem, A., & Fahmy, H. A. H. (2012).
LCAP - a lightweight CAN authentication protocol for securing in-vehicle networks. In 10th ESCAR conference on embedded security in cars.

[9] Kleberger, P., Olovsson, T., & Jonsson, E. (2011).
Security aspects of the in-vehicle network in the connected car. In Intelligent vehicles symposium (IV), 2011 IEEE (p. 528-533).

[10] Matsumoto, T., Hata, M., Tanabe, M., Yoshioka, K., & Oishi, K. (2012).
A method of preventing unauthorized data transmission in controller area network. In Vehicular Technology Conference (VTC Spring), 2012 IEEE 75th (p. 1-5).

[11] Ivan Studnia, Vincent Nicomette, Eric Alata, Yves Deswarte, Mohamed Kaaniche, et al.. Survey on security threats and protection mechanisms in embedded automotive networks. 2nd Workshop on Open Resilient Human-aware Cyber-Physical Systems (WORCS-2013), co-located with The 43rd Annual IEEE/IFIP International Conference on Dependable Systems and Networks (DSN-2013) (p. 1–12), Jun 2013, Budapest, Hungary. 2013, <10.1109/DSNW.2013.6615528>.<hal-01176042>.

[12] Wolf, M., Ruhr University Bochum at the department of Embedded Security (2009).
Security engineering for vehicular IT systems: improving the trustworthiness and dependability of automotive it applications. Springer.

[13] Hartkopp, O., Reuber, C., & Schilling, R., ESCAR, Berlin (2012).
MaCAN - Message Authenticated CAN. In 10th ESCAR conference on embedded security in cars.

[14] Moti Markovitz, Prof. AvishaiWool, ESCAR, Cologne (2015).
Field Classification, Modeling and Anomaly Detection in UnknownCAN Bus Networks.

[15] Timo van Roermund, ESCAR, Munich (November 17, 2016).
Securing the in-vehicle network trends, challenges, and solutions. In 2016 ESCAR conference.

[16] ROFIN-SINAR Technologies, Inc., Laser Image, accessed Dezember 27th, 2016
URL: http://www.rofin.de/m/de/assets/tabimages/other-rofin.png

[17] Nissan GT-R Nismo Now Available in Gran Turismo 6, accessed Dezember 27th, 2016
URL: http://www.motorward.com/2014/09/nissan-gt-r-nismo-now-available-in-gran-turismo-6/

[18] ELECTRONIC BIKE DERAILLEUR, accessed Dezember 27th, 2016
URL: http://hackaday.com/2012/06/06/electronic-bike-derailleur/