DoS Attacks on Controller Area Networks by Fault Injections from the Software Layer

Pal-Stefan Murvay
Politehnica University of Timisoara
Romania
pal-stefan.murvay@aut.upt.ro

Bogdan Groza
Politehnica University of Timisoara
Romania
bogdan.groza@aut.upt.ro

ABSTRACT

The Controller Area Network (CAN) is still the most widely employed bus in the automotive sector. Its lack of security mechanisms led to a high number of attacks and consequently several security countermeasures were proposed, i.e., authentication protocols or intrusion detection mechanisms. We discuss vulnerabilities of the CAN data link layer that can be triggered from the application level with the use of an off the shelf CAN transceiver. Namely, due to the wired-AND design of the CAN bus, dominant bits will always overwrite recessive ones, a functionality normally used to assure priority for frames with low value identifiers. We exploit this characteristic and show Denial of Service attacks both on senders and receivers based on bit injections by using bit banging to maliciously control the CAN transceiver. We demonstrate the effects and limitations of such attacks through experimental analysis and discuss possible countermeasures. In particular, these attacks may have high impact on centralized authentication mechanisms that were frequently proposed in the literature since these attacks can place monitoring nodes in a bus-off state for certain periods of time.

KEYWORDS

Controller Area Network, DoS, fault injection, bit banging

ACM Reference format:
Pal-Stefan Murvay and Bogdan Groza. 2017. DoS Attacks on Controller Area Networks by Fault Injections from the Software Layer. In *Proceedings of ARES '17, Reggio Calabria, Italy, August 29-September 01, 2017,* 10 pages.
DOI: 10.1145/3098954.3103174

1 INTRODUCTION

The need of connecting various system components by both cost-efficient and reliable communication mechanisms is a natural consequence of the rapid evolution in the complexity of embedded systems. This led to the development of a wide range of communication protocols that implement specific requirements of different industry sectors. The Controller Area Network (CAN) is one of the most prolific protocols and is established as the most widespread standard in the automotive industry. Besides its initial goal for building in-vehicle networks, it was also adopted in other industry areas such as avionics, automation and notably as part of demanding applications such as the CERN particle accelerator [3].

Due to its lack of security and the safety critical nature of the applications in which it is used, CAN attracted active interest from the security research community. One of the first experiments on sniffing and replay attacks on CAN buses came from Hoppe and Dittman [8] which proved the feasibility of such attacks on an electric window lift. The extensive experimental analysis performed on real-world automotives by Koscher et al. [14], Checkoway et al. [2] and Miller and Valasek [16] conclusively showed that CAN provides an easy to exploit surface for tampering with the systems interconnected by it and even paves way for deploying remote attacks with the aid of malicious devices planted on the bus. More recently Miller and Valasek [17] proved that remote attacks can be mounted without vehicle alteration by compromising CAN nodes capable of wireless communication. All these results along with the reported real-life incidents made it clear that the lack of security of the CAN protocol should be mitigated.

A number of solutions for securing the Controller Area Network were proposed as a result of the increasing number of reported attacks. Various approaches were considered, most of the efforts being directed towards application layer authentication mechanisms. Szilagyi and Koopman proposed a voting scheme [24] for time-triggered networks while another approach [6] proposes the use of a TESLA-like protocol, well known in sensor network applications, to provide delayed broadcast authentication. The LiBrA-CAN protocol proposes a Message Authentication Code (MAC)-based scheme which uses keys shared between groups of nodes [7] and authentication tags are mixed to increase security. Other approaches were centred on the physical layer proposing solutions like modifying the standard CAN message transmission to carry authentication information [25] or identifying nodes based on their unique signal characteristics [19]. Nonetheless, a centralized authentication scheme is explored in [15] where a master node discards malicious frames by overwriting them with error flags. This approach is particularly interesting as being both convenient for implementation and fully backward compatible.

In this work we investigate a layer of the CAN protocol stack which was mostly neglected from the security point of view. Namely, we focus on the security of the CAN data link layer which provides the means to mount Denial of Service (DoS) directly from the software layer without the need for complex electronics. These attacks can be mounted from the application layer by systems equipped with a basic CAN transceiver by fault injection using a bit banging technique. We present several attack variants and demonstrate their

ARES '17, Reggio Calabria, Italy

DOI: 10.1145/3098954.3103174

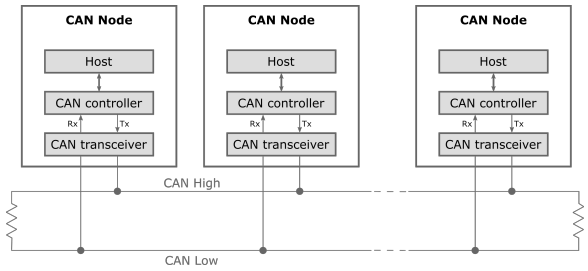

Figure 1: CAN bus topology

feasibility at speeds up to 1 Mbit/s by showing an experimental evaluation and discussing the requirements for mounting them.

The rest of this paper is structured as follows. In section 2 we present some background information on the CAN protocol which is needed to understand its vulnerabilities and further discuss related work. We proceed with the description of the attacks in section and present experimental results in section 4. Section 5 discuses possible countermeasures before concluding the paper in the final section.

2 BACKGROUND

2.1 The CAN protocol

CAN was introduced as a result of the stringent need in the automotive sector for a serial communication system at a time when the number of point-to-point connections required by the system architecture greatly increased the wiring overhead. Its development started in 1983 at Bosch which has published several versions of the protocol since then. The latest of these versions was published in 1991 and is known as CAN 2.0 [23]. Currently the CAN protocol specification is available as the multi-part standard ISO 11898. Part 1 [9] of the standard covers the CAN data link layer while parts 2 [10] and 3 [11] cover the physical layer for high-speed and low-speed CAN respectively. A number of CAN-based higher layer protocols were developed to fulfil the needs of various industrial sectors, e.g.: CANopen for industrial automation, SAE J1939 for commercial vehicles, UDS (Unified Diagnostic Services) for automotive diagnosis or NMEA 2000 for marine applications. While each of these higher layer protocols has its particularities, the underlying physical and data link layers of CAN are common to all.

CAN was designed as a two wire differential bus that interconnects a number of nodes. Each node that participates in CAN communication requires a CAN interface which is comprised of a CAN controller and a CAN transceiver connected by 2 wires as suggested in Figure 1. The CAN controller unit can be found as a stand-alone circuit or often as a dedicated module of the host microcontroller. The controller implements the CAN protocol at the data link layer as described in CAN 2.0 or ISO 11898-1 generating the transmit bit sequence or decoding incoming messages. The CAN transceiver is responsible with converting between logical data and the corresponding physical signalling as it connects the CAN controller to the physical communication lines. Depending on whether high-speed or low-speed CAN is employed, appropriate high- or low-speed transceivers have to be used because of differences in the signalling behaviour. High-speed CAN transceivers support bit rates of up to 1 Mbit/s while low-speed transceivers can only provide communication speeds of up to 125 Kbit/s. An advantage of the low-speed CAN transceivers is that they can provide fault tolerant communication.

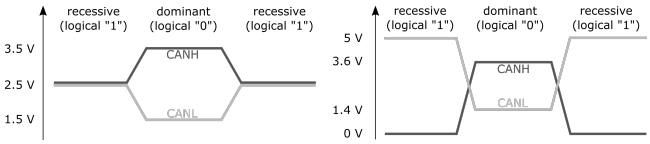

Figure 2: Bus levels for CAN high-speed (left) and low-speed (right)

The CAN specification defines the logical bit values as *dominant* when referring to logical "0" and recessive when referring to logical "1". These logical values are encoded for transmission on the physical layer as differential signals. The specific differential voltage levels are dependant on the type of transceiver as illustrated in Figure 2. A high-speed CAN transceiver (ISO 11898-2) interprets a differential voltage of up to 0.5 volts as a recessive value, while a differential voltage that exceeds 0.9 volts is considered as the dominant level. Low-speed transceivers (ISO 11898-3) will consider a differential voltage of 5 volts as a recessive bit and a typical differential voltage of 2 volts as a dominant bit. According to the CAN specification a dominant bus level will always overwrite a recessive bus level, therefore, the CAN bus is implemented as a wired-AND bus. This behaviour enables the implementation of CAN protocol features such as the arbitration-based bus access where the lowest message identifier has priority. The attacks presented in this paper are also facilitated by this behaviour.

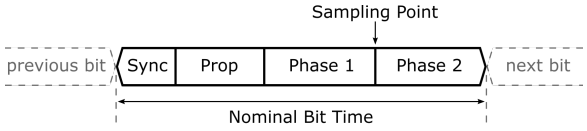

Figure 3: CAN bit timing

Figure 3 depicts the four segments that form a nominal bit. The length of each segment is an integer multiple of the Time Quantum (TQ), the smallest timing resolution used by a node to derive the bit time. The synchronization segment is where the transition from recessive to dominant or dominant to recessive is expected to occur. It is followed by the propagation segment introduced to compensate for the signal propagation delays. The Phase 1 and Phase 2 segments are used for resynchronization by being lengthened or shortened. The bit value sampling occurs directly after the Phase 1 segment. Hard resynchronizations occur on all recessive to dominant transitions signalling the beginning of a frame, i.e. the Start Of Frame (SOF) bit. Subsequent falling edges are used for soft synchronization which can only adjust the bit time by an amount specified as the Synchronization Jump Width (SJW) which is limited to a maximum of 4 TQs.

Data is sent over the CAN bus after being framed by the CAN controller. Besides the actual payload which can be 8 bytes at the

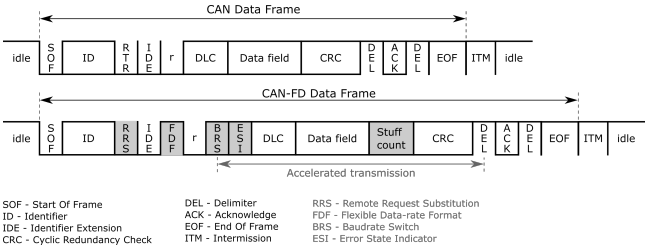

Figure 4: Standard CAN and CAN-FD data frame

most, each CAN data frame contains additional fields as depicted in Figure 4. The start of a data frame is signalled by the transmission of a dominant SOF bit (the idle state of the bus is represented by a recessive level). This bit is followed by the identifier field which has a length of 11 bits in standard frames and 29 bits in extended frames. In what follows we will be presenting the fields of a standard frame. The RTR bit comes next indicating if the frame contains actual data or it is just a remote request for data and the IDE bit which specifies the standard or extended type of the frame. After another bit which is reserved and should be set to the dominant state comes the data length code (DLC) field containing the length of the data field followed by the data field itself and a CRC field. A one bit CRC delimiter field is inserted after the CRC and should always be set to the recessive state. The 1 bit acknowledge field is also followed by a recessive delimiter before the 7 recessive bits of the EOF (end of frame) field.

CAN introduces an arbitration mechanism that assures collision avoidance on the bus. This mechanism is applied over the first part of the frame which includes the frame ID and requires the sending nodes to monitor the bit by bit transmission. If two or more nodes start sending a frame at the same time they each continue the transmission as long as the value of the bit read out from the bus equals the value they have written on the bus. Therefore, a node putting a recessive bit on the bus will always lose arbitration to a node writing a dominant bit, consequently identifiers with lower values have higher priorities.

Another mechanism employed by CAN is the bit stuffing procedure required to keep the nodes synchronised. The mechanism involves inserting additional bits of opposite value after each set of 5 identical consecutive bits. The stuffing bit will be inserted even if the 6th bit in the normal transmit sequence is different in value that the previous 5 identical bits.

2.2 CAN error management

Understanding the error management system from the CAN bus is relevant for understanding the attacks that we discuss. The CAN protocol defines an error detection mechanism based on bus monitoring performed by both the sender and receiver of a message. The sender is responsible with bit by bit monitoring of the sent message as well as the acknowledge field. By monitoring the bus bit levels the sender compares the sent bit value with the actual sampled bit value. A bit error exist if these values differ. Since all sent CAN messages should be acknowledged by a receiver node, the ACK field is checked by the sender. If the positive acknowledgement is missing, then the sender records an acknowledgement error. The message format, bit stuffing and checksum are verified on the receiver side. A form error occurs whenever a message is found to be non-conformant with the specification. Breaking the bit stuffing rules results in a bit stuffing error while failing to verify the message CRC produces a CRC error.

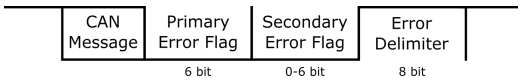

Figure 5: CAN error frame

Whenever an error is detected, the detecting node begins sending an error frame beginning with the first bit following the error detection. The CAN error frame, as illustrated in Figure 5, starts the Error flag which consists of 6 dominant bits. Even if the error was detected by a single node this field is meant to ensure that all other nodes send an error flag, e.g. as a result of the detection of a bit stuffing error. The Secondary error flag, also of dominant level, is meant to compensate for the later detection of errors and can be 0 to 6 bits in size. The last segment of the CAN error frame, called the error delimiter, consists of 8 recessive bits. After the error frame was sent and the intermission time has elapsed, the sender of the erroneous message tries to retransmit it.

The CAN protocol specification describes a fault confinement mechanism to prevent faulty nodes from creating high bus loads. According to this mechanism each node should implement two error counters: TEC (Transmission Error Counter) and REC (Receive Error Counter) These error counters are decremented by one on each successful transmission or reception of a data frame. Upon detection of an error, the sender node increments TEC by 8 while receivers increment REC by 1 unless they are the ones causing the error, in which case REC is incremented by 8. Depending on the values of these error counters a CAN node can be in one of three error states:

- **Error Active**. When in this state the CAN node behaves normally without any specific restriction
- **Error Passive**. Nodes in the Error Passive state can only indicate an error by sending 6 recessive bits preventing other nodes from globalizing the error. When sending consecutive data frames, nodes in this error state must wait for an additional time equivalent to 8 bits (Suspended Transmission Time).
- **Bus Off**. Nodes that reach the Bus off state can no longer influence the bus communication in any way. This state can only be exited after 128×11 correctly recorded recessive bits.

Figure 6 shows the possible transitions between these three states along with the triggering conditions.

2.3 CAN with Flexible Data-rate

The increasing demands for higher bandwidth have pushed the standard CAN to its limits leading to the need for developing an alternative that solves the bandwidth problem. Although a higher bandwidth alternative already existed, i.e. FlexRay, designing a cheaper alternative was of great interest. This is led to the design of CAN-FD (CAN with Flexible Data-rate), an extension to CAN

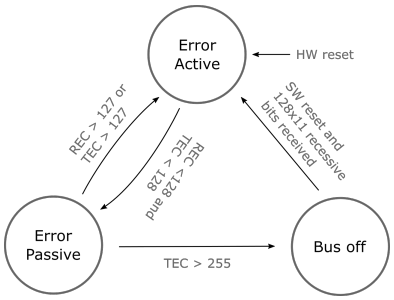

Figure 6: CAN error state transitions

that allows bigger payloads and higher bit rates for the payload. To accommodate the added functionality the standard CAN data frame is modified as illustrated in Figure 4. The RTR bit is renamed to RRS which is always dominant since the remote frame format does not need the FD extension. The reserved bit in the CAN frame is replaced with FDF which indicates a CAN-FD frame. The switch to a higher bit rate transmission is indicated by the BaudRate Switch (BRS) bit when set to a recessive state. The higher bitrate employed depends on the transceiver capabilities. Most of the existing CAN transceivers can cope with speeds of up to 5 Mbit/s. The high bandwidth segment ends at the sampling point of the CRC delimiter. Another feature introduced by CAN-FD frames is error state reporting which is done by setting the ESI bit to the dominant state when the node is in the Error Active state or to the recessive state when the node is in Error Passive. The 4 bits of the DLC field are used to encode payload sizes of up to 64 bits. In CAN-FD the bit stuffing rule is changed for the data field while the CRC field now accommodates 17 or 21-bit CRCs depending on the payload size.

2.4 Related work on bit-injection in CAN

The idea of mounting attacks by targeting the CAN protocol at the data link layer is not new. In [26] the authors mention the possibility of exploiting the CAN fault confinement mechanism to disconnect can nodes by sending well directed error flags but without any practical instantiation of the attack. On the practical side, injecting faults on the CAN bus has been in the focus of the automotive industry for setting-up network reliability tests, therefore special tools were designed for this purpose. One example of such a device is the Vector CANstress [5] which is capable of monitoring the CAN bus and injecting dominant and recessive bits in specified locations of targeted CAN frames as well as inducing a set of disturbances on the physical CAN layer. The CANstress or other similar devices could be used to fault injection attacks, however, their high price and bulky construction makes them inappropriate for a low cost stealthy attack.

More recently Palanca [20] demonstrated how a fault injection attack can be mounted on the parking sensor functionality of a 2012 Alfa Romeo Giulietta using an Arduino and a CAN transceiver. Their attack consists of leading a node in the bus off state by overwriting recessive bits with dominant bits to generate errors in target messages which were previously recorded by sniffing the CAN bus accessible through the vehicle diagnosis port. However, their results contain no analysis of the real limits of such attacks depending on the CAN bitrate (their experiment was done on a 50Kbit/s bus which is low considering the top CAN speed of 1 Mbit/s) or the possible ramifications of such attacks.

Another use for bit injection was in designing a CAN protocol variant which supports higher bit rates. The authors of CAN+ [27] proposed injecting additional bits at a higher frequency during each CAN bit time to reach up to 16 times the normal data rate.

3 ATTACK DESCRIPTION

3.1 Basic rationale behind the attacks

In order to mount an attack on the CAN data link layer one must be able to interfere with the logical data transmitted, implying that the attacker has the ability to affect the physical representation of bits on the bus. Given the intrinsic wired-AND nature of the CAN bus, any sampled value will be the result of the bus levels produced by all active bus nodes, i.e. a dominant state is recorded if at least one node outputs this state, while a recessive state is obtained only when all the nodes are producing a recessive output. Thus, by using a CAN transceiver to generate the physical layer signals along with a custom-built CAN controller one would be able to design a CAN node that may force bits into the dominant state at will. A custom CAN controller is required since standard controllers are build to comply with the CAN specification and cannot be used to produce non-compliant behaviour. An efficient CAN controller implementation could be FPGA-based and the transceiver could also be replaced with dedicated circuitry that allows forcing both dominant and recessive bits. However, here we focus on mounting attacks with off the shelf components found in any CAN node, i.e., a CAN transceiver and a microcontroller.

A microcontroller can interact with the transceiver connected to it directly through its I/O ports without the need of the CAN controller. The application layer has the ability to interact directly with the CAN transceiver due to the nature of CAN transceivers which are only responsible of translating logical levels provided on the Tx pins to physical bus levels and providing feedback on the existing bus levels on the Rx pin in a digital form. This makes it possible for any electronical circuit to interract with the transceiver as long as it is capable of generating the needed logical Tx levels and reading back the Rx line. Therefore, the controller logic can be implemented in software making it feasible for any electronic unit equipped with a transceiver which is directly connected to a microcontroller to monitor the communication and inject bits on the bus. This way of controlling serial communication through software instead of hardware is known as bit banging and is often employed for implementing some communication protocols with minimal hardware requirements or for inducing protocol violations. The only limitation is given by the bit duration which constraints the amount of software computation that can be performed for each bit. The setup needed for this attack is presented in Figure 7. An attacker can accomplish it by either introducing a self-built device on the bus or by compromising an existing node by reprogramming it (recently it was shown by [17] that reprogramming nodes might be even done remotely).

Using this setup and the ability to inject dominant bits at specific locations inside the frame, several attacks can be deployed. We present these attacks in the following section.

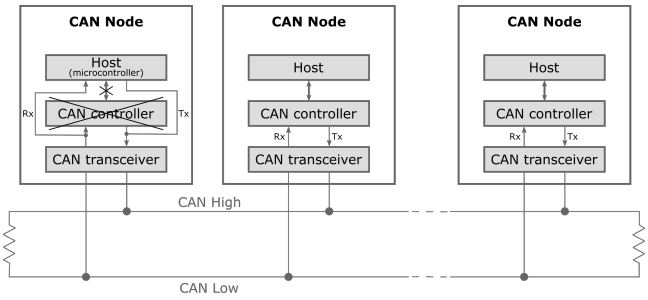

Figure 7: Attack setup

3.2 Attack variants

Full bus DoS. The CAN communication can be completely prevented by generating a continuous dominant state on the bus. This will prevent any other node from sending any type of messages on the bus. To achieve this, an attacker should assure a stable "0" level on the Tx line of the CAN transceiver. Many CAN transceivers have built-in mechanisms for preventing bus disturbance caused by an unintentional short to ground of the Tx line. The protection circuit will disable the transceiver output drivers when the length of the Tx dominant level exceeds a specified duration. Therefore, additional actions should be taken by the attacker to overcome this issue. In most cases a simple toggling of the Tx line at time intervals smaller than the error detection threshold is sufficient. Some CAN transceivers such as the MC33742 system basis chip implement more advanced error confinement mechanisms which require a specific reset command to re-enable the output drivers. This downfall can be surpassed by monitoring the transceiver error state and resetting it as soon as detected.

Directed DoS. The DoS attack can be directed to a single node on the bus by injecting dominant bits only in messages sent by the target node. For this, the attacker should have a-priori knowledge on the IDs of the messages sent by a certain node. Then by monitoring the bus for the target IDs it should inject dominant bits instead of recessive bits following the arbitration phase. Replacing recessive bits with dominant ones in the arbitration phase would result in the target ending transmission due to arbitration loss but this is not the intent of this attack, hence we discuss it as a distinct usecase. Following the bit value manipulation the sender node will detect a bit error and immediately start sending an error frame and increment its transmission error counter TEC by 8. By similarly affecting subsequent retransmission requests the TEC of the targeted node will exceed 255 forcing the node to enter the Bus Off state. Depending on the local Bus Off handling the target node might stay disconnected until a hard reset or re-enter the Error Active state after the detection of 128×11 recessive bits. In this second case the attacker can continue preventing successful message transmission from the target node using the described approach.

The time needed to make a bus go into Bus Off will depend on several factors such as the bitrate used, the location within the message where the fault is injected and the presence of other higher priority bus traffic. Figure 8 illustrates the influence of bitrates higher than 100 kbit/s and fault injection location over the time needed for a node to go into Buss Off considering a target message with an 8 byte payload sent over a bus without any additional traffic. The higher the employed bitrate the faster the Bus Off state is reached, e.g. at 1 Mbit/s it will take at most $4.9ms$ before the target node reaches the Bus Off state while at 125 kbit/s the node will enter Bus Off after a maximum period of $39,17ms$. To accelerate the process the bit injecting should occur as early as possible in the frame transmission slot.

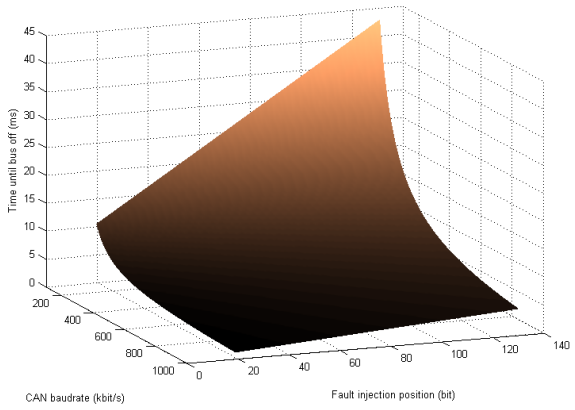

Figure 8: Time needed to force a node to the Bus Off state depending on the bit rate and fault injection location within the frame (with 8 byte payload) for bitrates higher than 100 kbit/s in the absence of other bus traffic

Arbitration denial. As mentioned in the previous attack, by injecting dominant bits in the frame arbitration area an attacker could prevent a node from winning arbitration and therefore, preventing it to access the bus. As a side-effect this attack could also stop messages with IDs which are not targeted depending on the position of the injected dominant bit within the arbitration field. Once the target node has stopped transmitting due to arbitration loss the attacker can release the dominant bus state. Since the frame is not complete, an error (e.g. bit stuffing or form error) will be detected by all nodes on the bus and reported through an error frame. This will, however, not result in any node entering the Bus Off state because all nodes will simultaneously enter the Error Passive state at which time the error counters will not be increased past the threshold due to the lacking dominant error flag (Error Passive nodes are not allowed to send dominant error flags). To force all nodes in the Bus Off state, the attacker should also inject the dominant error flag in the error frame space. Performing such an attack with the intent to prevent a certain message to reach the bus would be more time consuming since the message retransmission by the targeted node would occur faster than in the case of a fault injection during or past the data field. Also, forcing the Bus Off state will take longer since for each interrupted transmission the TEC will be incremented by 1.

Disrupting synchronization. Synchronization between a sender node and the receivers is assured by a hard clock synchronization done at the start of frame and subsequent limited clock adjustments done at every transition from recessive to dominant. Therefore, by placing a falling edge followed by a transition back to the recessive

state in the propagation segment of a recessive bit (this should occur before the sampling point to avoid affecting the sampled bit value) an attacker could determine the receivers to re-adjust their bit time. This may have one of two outcomes, depending on the bit timing settings of each network node. If all nodes have identical or close enough bit settings, injecting a pulse during the bit time will result in a change of the bit duration and, therefore, each attacked frame will generate a higher or lower bus load. Otherwise, if bit timing settings and the SJW (i.e. the maximum bit time adjustment amount) in particular are different throughout the network the constant pulse injection will result in de-synchronisation and subsequently communication errors will occur.

Figure 9 illustrates the lengthening of a bit by injecting a falling edge in the propagation field. This bus manipulation would not be detected by the sender node as a bit error since it does not check the bus level over the entire bit time for synchronization but only does so on the sampling point by which time the attacker has switched back the bus level to its original value.

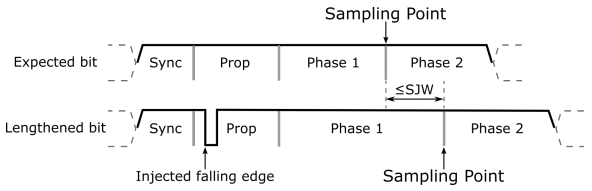

Figure 9: Intentionally lengthened CAN bit timing attack

The amount of lengthening is controlled by modifying the distance between the bit synchronization segment and the injected pulse but cannot exceed the SJWs employed by the CAN controllers. According to the CAN speciffication the SJW cannot exceed 4 TQs. The TQ configured by each CAN node will also influence the maximum delay which can be induced. The SAE J2284/3 standard [21] makes recommendations for configuring bit timing for high-speed 500 kbit/s communication. Considering these recommendations for a standard CAN frame with an 8 byte data field in which 50% of the bits are recessive and there are no stuffing bits the maximum theoretical increase in frame length would be between 6.15% and 9.01%, depending on the bit timing settings.

3.3 CAN DoS impact on system reliability

Any DoS attack on systems employing CAN communication would affect functionalities that rely on CAN communication to perform properly. This includes subsystems that require any kind of input being received over CAN from other subsystems. In automotive systems safety critical components, e.g., the engine control unit, can continue performing with limited functionality in the event of loosing CAN communication capabilities. This limited functioning regime is called *limp mode* and was designed to allow the car to be safely driven home or to the nearest service center. Therefore, when under a CAN DoS attack these safety critical systems will continue functioning but with limited functionality while all other vehicle systems not considered in the limp mode would be unable to completely fulfil their functionalities that require CAN communication. One example of a system that would be greatly affected by a DoS attack is the cluster instrument panel which displays information on a wide range of car subsystems. Preventing CAN status messages from being sent to the cluster instrument would render it useless since it would not be able to display information on the actual vehicle state.

3.4 Impact on proposed authentication protocols for CAN

We give some brief clarifications on the impact of bit-flip attacks on proposed security protocols for CAN. In principle, bit injection leads to two kinds of attacks: i) placing nodes in a bus-off state due to accumulation of send/receive errors and ii) changing the values of the interpreted messages in case when the error control mechanisms in CAN is bypassed (e.g., [18] and [12]).

The first type of attack has impact on centralized authentication schemes were the authentication master can be put off-line. In case of the protocol proposed in [15], if the master is placed in bus-off, forged frames that are injected by an intruder will not be discarded by error flags and will pass as authentic. This leads to breaking authenticity. For other protocols where authentication is done in a centralized manner, the master node will not be able to send the authentication tag and in this case the frame will be discarded by the receivers. This is simply a DoS attack and may happen on protocols such as the centralized authentication version of Libra-CAN [7]. Bit injection may also influence protocols such as CAN-Auth [25] in a distinct fashion. CAN-Auth [25] takes advantage of the CAN+ extension to CAN [27] which injects additional bits before the regular sampling point of a CAN bit. In this way, regular CAN frames are unchanged by CAN-Auth and the additional security bits are inserted before the sampling point. However, by bit injections, the authentication bits can be erased and genuine frames will have normal data fields while the authentication bits will be altered. Moreover, as discussed in section 4.2, bit injection may force the nodes to adjust the sampling point, if the sampling may be forced inside the CAN+ transmission window CAN+ sender nodes will be put in bus off by error flags from regular nodes which will detect errors in the data field. Validating such attacks on CAN+ is out of reach for the current work since CAN+ is only available as prototype FPGA implementation.

Negotiation of wrong cryptographic keys is possible for proposals such as the ones from [18] and [12]. The nodes will detect that they are in the possession of the wrong key and the adversary cannot control the bits of the key so the attack is again simply a DoS and keys will be renegotiated. We point out that the DoS is indeed a less relevant adversarial action and placing the authentication master in a bus-off is a more dangerous consequence. We summarize these potential threats in Table 1.

4 EXPERIMENTAL ANALYSIS

4.1 Experimental setup

Our experimental setup consisted of 2 or 3 node CAN networks out of which one was the attacker node. We implemented the attacker capabilities on platforms equipped with 16-bit microcontrollers from the S12X family, namely the S12XD512 and S12XF512 family members. The choice for these microcontrollers is motivated by their common usage in the automotive industry as well as in

Table 1: Potential threats from bit-injection to existing CAN security proposals

Protocol	CANAuth'11	LiBra-CAN'12	Kurachi et al.'14	Mueller & Lothspeich 15	Jain & Guajardo'16
DoS at authentication	✓	✓	-	-	-
DoS at key-exchange	-	-	-	✓	✓
Loss of authentication	-	-	✓	-	-

other industrial applications. Equipped with up to 32 Kbytes of RAM, 512 Kbytes of Flash and running at maximum frequencies of 50 MHz, these microcontrollers are representative for the low- to mid-end performance products which is another reason for choosing them for setting a baseline for performing the described attacks. We evaluated the attacks on both high-speed and low-speed CAN buses by using two sets of setups, i.e. one for each category of transceivers. On the attacker side we used the TJA1054T as the low-speed can transceiver and the MC33742 system basis chip which integrates a high-speed CAN transceiver. As the attack target nodes we employed similar platforms as well as other off-the-shelf devices such as USB-to-CAN devices equipped with PCA82C251 high-speed CAN transceivers and VN7570. Figure 10 shows one of the experimental settings employed for testing attacks on high-speed CAN nodes featuring two S12XF512-based development boards (one as the attacker, the second as a legit node), an USB-to-CAN and a Picoscope used for analysing the physical signalling during the attack.

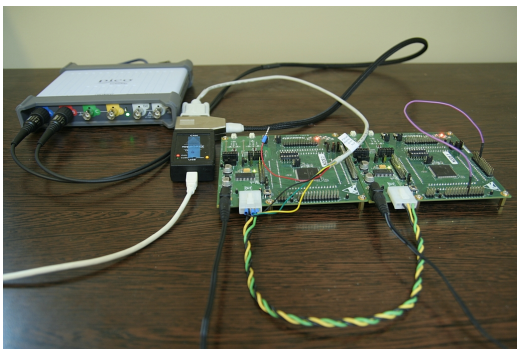

Figure 10: Experimental setup used for testing attacks high-speed CAN

4.2 Attack validation

Full bus DoS. Mounting this attack is generally straightforward and does not introduce a considerable increase of the CPU load. The port pin connected to the CAN transceiver's Tx line is configured as an output pin and set to "0" for the duration of the attack. Also, depending on the transceiver type, additional periodical toggling of the Tx line to "1" may be required to reset the transceiver internal fault detection timers. Both the low- and high-speed transceivers employed to build our attacks included mechanism for detecting permanent dominant errors on the Tx pin. Therefore, we implemented periodic toggles of the Tx line to the recessive state and back. In each case, the duration of the recessive pulse was determined empirically as the smallest value detectable by the transceiver. With the attacker node added to the bus no other CAN node was able to send messages for the entire attack duration. Te effect was the expected one since by forcing the lines to remain in the dominant state no other signalling is possible due to the wired-AND behaviour of the CAN bus.

Directed DoS. Directed attacks are aimed at specific frames whether this means targeting all messages with a specified ID or a precisely defined message. Performing this kind of attacks requires the ability to compare each bit on the bus against the defined filter and injecting dominant states to the specified locations while fitting the frame bit timing. Our goals were to investigate the feasibility of directed fault injection DoS attacks on various CAN bus types (i.e. low-speed CAN, high-speed CAN and CAN-FD) as well as performance-related limitations in mounting them.

On the low-speed CAN setup, which is limited to speeds of up to 125 kbit/s, we were able to implement a more advanced software bit monitoring mechanism including identifying individual CAN frame fields while using the maximum 50 MHz microcontroller clock frequency. We used this implementation to successfully implement the attack on messages sent with the top 125 kbit/s bitrate. Figure 11 illustrates the bus levels while sending an unaffected frame and while a dominant bit is injected at the start of the data field in the same frame. As expected, the bit injection causes error frames to be sent by the node sending the original message and 31 retransmit attempts after which the node goes into the Bus Off state which it holds for $11.264ms$, i.e. 11×128 recessive bits at 125 kbit/s (in the absence of other bus traffic).

For high-speed CAN, we aimed at implementing the attack for the top 1 Mbit/s CAN communication speed. Given the $1\mu s$ bit time constraint we were only able to build the attack for messages with precisely specified bit sequences. This means that the knowledge of exact frame representation on the bus including stuffing bits is required for performing the attack while using 1 Mbit/s bit rates. This limitation is given by the platform employed in our experimental setup, therefore, less constraints should be obtained by using devices with higher operating frequencies and more efficient CPU architectures. For speeds up to 500 kbit/s we were able to implement more flexible versions of the attack. Using the same high-speed implementation we also successfully performed the attack on CAN-FD frames which are using a higher bitrate for the data field. Figure 12 illustrates our attack mounted on an CAN-FD frame transmitted using 5 Mbit/s as the higher bitrate and 1 Mbit/s for the other frame fields.

We compared the effect of our attack with the capabilities of the CANstress tool and found them to be very similar as depicted in Figure 12 for the attack on a CAN-FD frame. The only noticeable differences are in the signal form and timing. Even though in our attack the signal exhibits higher influences from the legitimate frame it does not fail in keeping the line voltage to a dominant level.

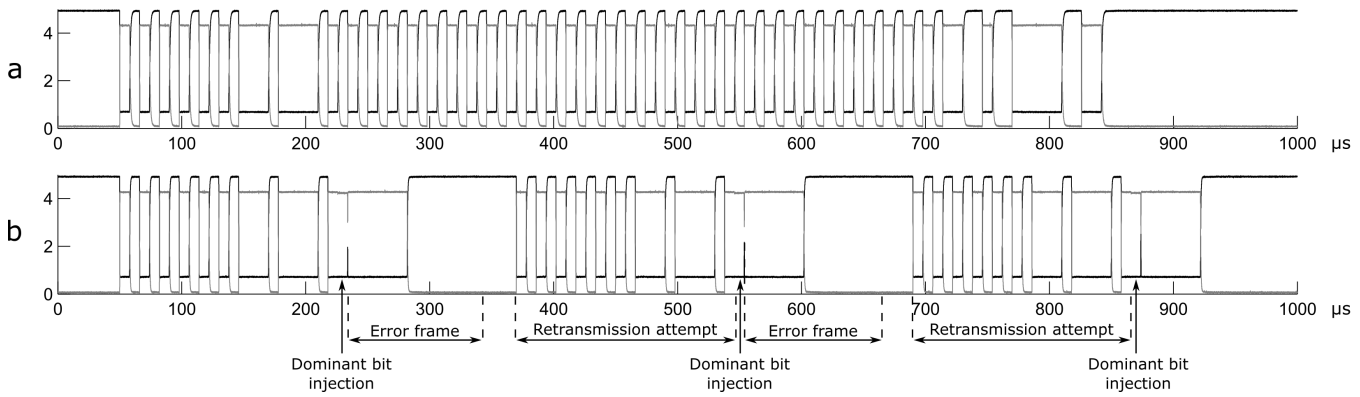

Figure 11: Directed DoS on a CAN frame sent over a Low-speed bus at 125 Kbit/s: a - original frame, b - attacked frame with error signalling and retransmission attempts

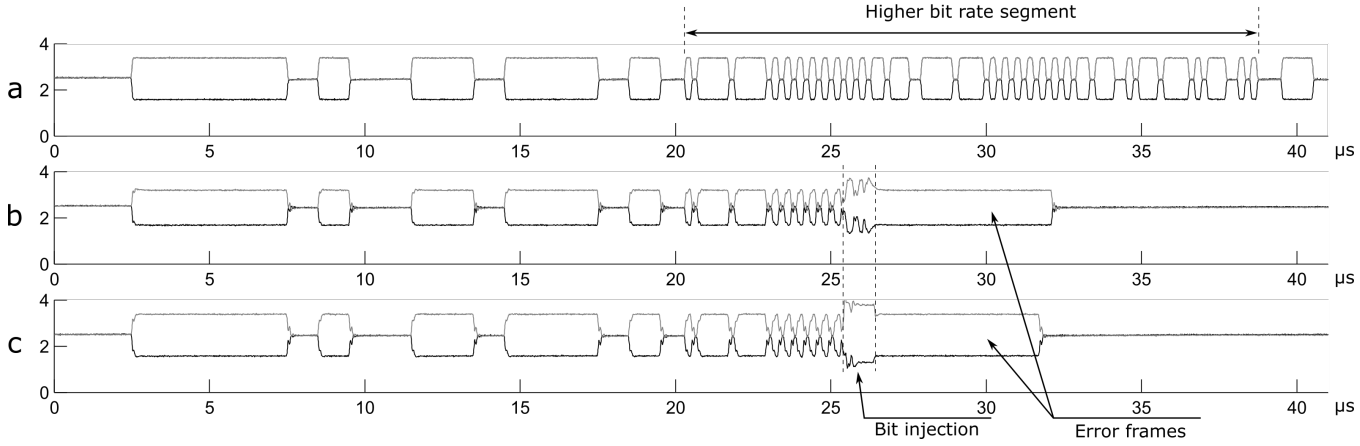

Figure 12: Directed DoS on a CAN-FD frame sent at 1 Mbit/s with 5 Mbit/s data rate: a - Original frame, b - Frame with SW implemented bit injection in the data field, c - Bit injection on the same position with the CANstress device

The CANstress is more precise in timing its bit injection due to the superior performance. For speeds around the 1 Mbit/s limit our fault injection timing is off by several tens of nanoseconds due to the inherent device limitations but are still sufficiently precise to disrupt communication at the intended bit location.

Arbitration denial. The actual implementation of the arbitration denial attack is similar to the approach used for the Directed DoS attack. The only significant difference is the fact that the fault injection occurs inside the arbitration field of the target frame. We tested the attack on both setups and were successful in stopping the transmission of targeted frames by injecting a single dominant bit as a replacement for a recessive bit in the arbitration field. If the attacker does not send further bits a form or bit stuffing error will be detected by all nodes on the bus and an error frame will be issued before the target node retries the transmission. After a while all nodes enter the error passive state and are not permitted to generate a dominant error flag any more. Therefore, a 23 recessive bit interval (6 recessive bits to trigger the stuffing error + 6 bit recessive error flag + 8 bit error delimiter + 3 bit inter-frame time) will be seen between the transmission interruption and any other message transmission. No Bus Offs occurred as a result of this attack.

Disrupting synchronization. We performed a series of tests with resynchronization pulse injection on various bit timing configurations both for the case when all nodes have the same bit settings as well as the case when they are configured differently. We found that the ability to manipulate frame duration through forced resynchronization greatly depends on the attacker knowledge of the individual bit timing settings on each network node. Failing to comply with bit timing characteristics of all nodes led to de-synchronization which resulted in errors being reported. We found the attack can still be mounted without a priori information about the bit timing settings employed throughout the network by gradually changing the resynchronization pulse location until it begins generating error frames. The ability to mount this attack greatly depends on the computational power of the processor used to mount the attack since it should be able to generate pulses finely grained to the size of a time quanta. We were not able to successfully mount the attack on bit rates higher than 500 kbit/s using our S12X-based platforms.

5 COUNTERMEASURES

In the light of the reported attacks, we now briefly discuss on attack detection and countermeasures, i.e., prohibiting an adversary from performing the attack.

Detecting the attacks. The detection of such attacks could be carried out by using similar methodologies as in intrusion detection systems, i.e., by analyzing the network traffic and specifically the physical layer. Signal processing techniques such as the ones in [19] where the sender of each message is identified based on signal characteristics are promising for detection. As can be seen in Figure 11 the pattern of the attack signal is quite distinct from the regular one, thus detection should be readily achieved.

Implementing countermeasures is unfortunately not a straightforward task. The difficulty stems from the fact that an attacker node can only be prevented from performing the attack by blocking its access to the CAN network.

Enhanced electronics. When using a bus topology, identifying and blocking the attacker node might prove to be a difficult task. Assuming that a bus monitoring system capable of identifying the attacker node is in place it will require an additional mechanism to stop this node from further attacks. This mechanism could be implemented in the form of a smart fuse box commanded by a supervisor node to power-off any network node. However, this scenario would not be effective in the case of custom self-powered attacker nodes infiltrated in the network. In what follows, we present two potential countermeasures which are more efficient in handling this category of DoS attacks.

Active star topologies with network guardian. We envision a more efficient protective measure capable of eliminating the attack which requires the use of an active star network topology in which all nodes are connected through a Network Guardian node. Such an approach was considered for fault confinement in CAN networks [1] by Barranco et al. The Network Guardian should implement an intrusion detection mechanism based on CAN protocol misbehaviour. Since all communication in this star topology would pass through the guardian node, a node trying to mount any of the presented attacks would be easily detected since they involve deviations from the CAN protocol specification. Upon detection any further communication to and from the attacker node would be prohibited.

The obvious disadvantage of such an approach would be the increased wiring complexity since a separate connection would be needed between each node and the guardian. The wiring complexity could be reduced by partitioning the network around several guardian nodes according to the actual placement of the nodes within the vehicle.

Enhanced fault confinement mechanisms. Another approach to alleviate the effects of such attacks is to change the fault confinement mechanisms of CAN. Such a proposal came more than a decade ago when the authors from [4] after a careful analysis of CAN fault confinement mechanisms noticed that the bus-off state may be reached too easy. Their proposal is to take a bus-off decision only for nodes that jeopardize the real-time behaviour of the traffic. This would be a natural approach and could remove the DoS attacks on master-oriented authentication that was previously discussed. As already noticed in [4], such mechanisms can be easily deployed by bypassing the existing CAN fault-confinement mechanism and implementing a new one at the application layer.

6 CONCLUSIONS

Considerable effort has been put in evaluating the security of CAN-based networks and it revealed a number of attacks which can be easily performed given the lack of security mechanisms within the CAN specification. We add to these reports a new set of DoS attacks aimed at the Data Link Layer of CAN based on injecting dominant bits during frame transmission. These attacks can be mounted on any CAN-based protocol (e.g., CANopen, TTCAN, SAE J1939, etc.) using off the shelf components requiring only the ability to control a CAN transceiver's digital pins by an application layer attack logic. To successfully mount such an attack one must be able to infiltrate a device on the CAN bus of the target system or compromise one of the existing network nodes. The latter scenario is realistic especially in the automotive domain since attacks leading to successful remote reprogramming of in-vehicle nodes were reported [17].

The impact of this type of attacks is even more severe as they can also be used against some of the security mechanisms proposed for CAN buses rendering them unusable. Moreover, since they affect the CAN Data Link Layer, these attacks are harder or impossible to be detected by Application Layer intrusion detection mechanisms based on traffic analysis like [22] or [13] since these would not be able to distinguish between injected faults and an honest node with faulty behaviour. Due to the safety-critical nature of in-vehicle communication, these attacks should not be overlooked in real-world security deployments.

ACKNOWLEDGMENTS

This work was supported by a grant of the Romanian National Authority for Scientific Research and Innovation, CNCS-UEFISCDI, project number PN-II-RU-TE-2014-4-1501 (2015-2017).

REFERENCES

[1] Manuel Barranco, Julián Proenza, Guillermo Rodríguez-Navas, and Luís Almeida. 2006. An active star topology for improving fault confinement in CAN networks. *IEEE transactions on industrial informatics* 2, 2 (2006), 78–85.

[2] Stephen Checkoway, Damon McCoy, Brian Kantor, Danny Anderson, Hovav Shacham, Stefan Savage, Karl Koscher, Alexei Czeskis, Franziska Roesner, Tadayoshi Kohno, and others. 2011. Comprehensive Experimental Analyses of Automotive Attack Surfaces. In *USENIX Security Symposium*.

[3] D Di Calafiori, P Adzic, G Dissertori, Oliver Holme, Dragoslav Jovanovic, W Lustermann, and S Zelepoukine. 2012. Maintaining and improving the control and safety systems for the Electromagnetic Calorimeter of the CMS experiment. In *Journal of Physics: Conference Series*, Vol. 396. IOP Publishing, 012016.

[4] Bruno Gaujal and Nicolas Navet. 2005. Fault confinement mechanisms on CAN: analysis and improvements. *IEEE transactions on vehicular technology* 54, 3 (2005), 1103–1113.

[5] Vector Informatik GmbH. 2006. User Manual CANstress, Version 2.1.

[6] Bogdan Groza and Stefan Murvay. 2013. Efficient protocols for secure broadcast in controller area networks. *IEEE Transactions on Industrial Informatics* 9, 4 (2013), 2034–2042.

[7] Bogdan Groza, Stefan Murvay, Anthony Van Herrewege, and Ingrid Verbauwhede. 2012. Libra-can: a lightweight broadcast authentication protocol for controller area networks. In *International Conference on Cryptology and Network Security*. Springer, 185–200.

[8] Tobias Hoppe and Jana Dittman. 2007. Sniffing/Replay Attacks on CAN Buses: A simulated attack on the electric window lift classified using an adapted CERT taxonomy. In *Proceedings of the 2nd workshop on embedded systems security (WESS)*. 1–6.

[9] ISO. 2003. *11898-1–Road vehicles–Controller area network (CAN)–Part 1: Data link layer and physical signalling.* Technical Report. International Organization for Standardization.

[10] ISO. 2003. *11898-2, Road vehicles Controller area network (CAN) Part 2: High-speed medium access unit.* Technical Report. International Organization for Standardization.

[11] ISO. 2006. *11898-3, Road vehicles Controller area network (CAN) Part 3: Part 3: Low-speed, fault-tolerant, medium-dependent interface.* Technical Report. International Organization for Standardization.

[12] Shalabh Jain and Jorge Guajardo. 2016. Physical Layer Group Key Agreement for Automotive Controller Area Networks. In *Conference on Cryptographic Hardware and Embedded Systems*.

[13] Min-Joo Kang and Je-Won Kang. 2016. Intrusion Detection System Using Deep Neural Network for In-Vehicle Network Security. *PloS one* 11, 6 (2016), e0155781.

[14] Karl Koscher, Alexei Czeskis, Franziska Roesner, Shwetak Patel, Tadayoshi Kohno, Stephen Checkoway, Damon McCoy, Brian Kantor, Danny Anderson, Hovav Shacham, and others. 2010. Experimental security analysis of a modern automobile. In *Security and Privacy (SP), 2010 IEEE Symposium on*. IEEE, 447–462.

[15] R. Kurachi, Y. Matsubara, H. Takada, N. Adachi, Y. Miyashita, and S. Horihata. 2014. CaCAN - Centralized Authentication System in CAN (Controller Area Network). In *14th Int. Conf. on Embedded Security in Cars (ESCAR 2014)*.

[16] Charlie Miller and Chris Valasek. 2013. Adventures in automotive networks and control units. *DEF CON* 21 (2013), 260–264.

[17] Charlie Miller and Chris Valasek. 2015. Remote exploitation of an unaltered passenger vehicle. *Black Hat USA* 2015 (2015).

[18] Andreas Mueller and Timo Lothspeich. 2015. Plug-and-secure communication for CAN. *CAN Newsletter* (2015), 10–14.

[19] Pal-Stefan Murvay and Bogdan Groza. 2014. Source identification using signal characteristics in controller area networks. *IEEE Signal Processing Letters* 21, 4 (2014), 395–399.

[20] Andrea Palanca. 2016. *A Stealth, Selective, Link-layer Denial-of-Service Attack Against Automotive Networks.* diploma thesis. Politecnico di Milano.

[21] SAE. 2002. *High-Speed CAN (HSC) for Vehicle Applications at 500 KBPS.* Standard. SAE International.

[22] H. M. Song, H. R. Kim, and H. K. Kim. 2016. 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)*. 63–68.

[23] CAN Specification. 1991. *Version 2.0.* Technical Report. Robert Bosch GmbH.

[24] Chris Szilagyi and Philip Koopman. 2010. Low cost multicast authentication via validity voting in time-triggered embedded control networks. In *Proceedings of the 5th Workshop on Embedded Systems Security*. ACM, 10.

[25] Anthony Van Herrewege, Dave Singelee, and Ingrid Verbauwhede. 2011. CANAuth-a simple, backward compatible broadcast authentication protocol for CAN bus. In *ECRYPT Workshop on Lightweight Cryptography*, Vol. 2011.

[26] Marko Wolf, André Weimerskirch, and Christof Paar. 2004. Security in automotive bus systems. In *Workshop on Embedded Security in Cars*.

[27] Tobias Ziermann, Stefan Wildermann, and Jurgen Teich. 2009. CAN+: A new backward-compatible Controller Area Network (CAN) protocol with up to 16× higher data rates.. In *Design, Automation & Test in Europe Conference & Exhibition, 2009. DATE'09.* IEEE, 1088–1093.