Practical security exploits of the FlexRay in-vehicle communication protocol

Pal-Stefan Murvay[1] and Bogdan Groza[1]

Politehnica University of Timisoara, Timisoara, Romania
{pal-stefan.murvay, bogdan.groza}@aut.upt.ro

Abstract. The ever increasing number of electronic control units inside a car demanded more complex buses with higher bandwidth capacities. But even the more recently designed in-vehicle network protocols, e.g., FlexRay, were engineered in the absence of security concerns and thus they are highly vulnerable to adversarial interventions. In this work, we study the FlexRay protocol specification to identify features that can be used to mount various attacks. The attacks exploit both the physical layer and the data-link layer of the protocol to discard messages from the bus, i.e., DoS attacks, or to spoof messages by inserting adversarial frames and later discarding the genuine frames. We illustrate the feasibility of these attacks on an experimental setup composed of several FlexRay nodes implemented on automotive-grade controllers. While these attacks may not be a surprise, recognizing them may be relevant in preventing potential future exploits.

Keywords: security · FlexRay · attacks · DoS · automotive.

1 Introduction and motivation

Modern vehicles are complex systems integrating an ever increasing number of electronic control units (ECUs) interconnected via dedicated communication lines. While the most widely used in-vehicle communication bus is still the decades old CAN (Controller Area Network), FlexRay was designed as the bus for future automotive applications. Indeed, FlexRay supports high data rates of up to 10Mbit/s in contrast to the 1Mbit/s of the regular CAN bus and is deterministic in nature which makes it suitable for hard real-time applications, e.g., x-by-wire technologies that are replacing regular mechanical interfaces.

Denial-of-service (DoS) attacks are a common concern for computer networks. For in-vehicle networks however, these were generally neglected from the same reasons as security was ignored: in-vehicle networks were perceived as isolated from outsiders. Today we are aware that this is not so as recent research demonstrated attacks that can be carried out from the outside over in-vehicle buses, e.g., [4], [12].

In previous work we have already explored DoS attacks on the CAN protocol which were triggered by directly controlling the physical bus signaling from the application layer [13]. Similar DoS attacks on the CAN bus were also shown in [15] while the work in [5] only focuses on targeted DoS attacks by placing specific CAN nodes in the bus-off state. All these attacks exploit the CAN error handling mechanism and may have severe consequences as a targeted DoS attack on a specific node may further facilitate impersonation attacks. Such attacks are easy to mount on the CAN bus since the error confinement mechanism is built to isolate faulty nodes and reacts on malformed frames. Even at the time of our previous work it was apparent to us that similar attacks can be mounted on FlexRay but the denser protocol specification made it less obvious how easy is to deploy the attacks. As we later discuss in the related work section, attacks on the FlexRay network were suggested as early as the work in [16] and even demonstrated almost a decade ago by the authors in [14] via a FlexRay simulation in CANoe. But since then, these attacks appear to be somewhat neglected. We were able to find more recent results only in [7] where a man-in-the-middle adversary for FlexRay is discussed. We contribute to these by giving more practical insights and experimental results on DoS attacks and frame spoofing on FlexRay by using a Freescale (NXP) evaluation board. In Table 1 we give a summary of the attacks that are tested in our work, more details will be given in the forthcoming sections.

Table 1. Summary of potential attacks on FlexRay (tested on the Freescale (NXP) evaluation board)

Attack	**Static Segment**	**Dynamic Segment**
DoS-full	Yes, required adversary actions: i) place the bus in a dominant state (physical layer) or, ii) break synchronization by sending frames in occupied slots (data-link layer)	
DoS-targeted	Yes, required adversary actions: same as full DoS i) or ii) for specific frames	Yes* *only if frame occurrence is predictable
Msg. spoof	No* *collisions lead to unpredictable bus state	Yes*, required adversary actions: redefine messages in own comm. cycle * may result in collisions, i.e., targeted DoS

1.1 Related work

Security in automotive networks was intensely studied in recent years. Most of the research done in this area had CAN as the main focus since it is the most widely used communication technology for in-vehicle networks. CAN was proved to be vulnerable to replay and spoofing attacks which are easy to mount once access is gained to the in-vehicle network [11, 4, 12]. Moreover, due to its arbitration and fault confinement mechanisms CAN is susceptible to denial of service attacks (DoS) [13].

One of the first mentions of security issues regarding the FlexRay protocol were made by Wolf et al. which briefly mentions several potential attacks [16]. The first requires the exploitation of the bus guardian for disabling targeted nodes by sending faked error messages. This is not an immediate threat since the bus guardian specification is still preliminary and, to the best of our knowledge, there exists no bus guardian implementation in use. The second mentioned attack, which we investigate in more detail in our work, refers to impeding communication by causing loss of synchronization. Finally, the third proposed attack involves disabling nodes with power saving capabilities by sending malicious messages to force them enter sleep mode. No experiments are provided to demonstrate the attacks, but outlining them is valuable.

In the work done by Nilsson et al. [14] the FlexRay protocol specification is analysed in search of security mechanisms for providing basic security objectives. Such mechanisms are clearly not present in FlexRay since security was not considered when designing this protocol. Furthermore, they employ a simulated environment in CANoe to prove that mounting read and spoofing attacks on FlexRay is an easy task. As we discuss in section 3, this kind of attack is only possible in certain situations since otherwise it may result in collisions on the bus.

A more recent study [7] shows how an adversary listening to a FlexRay network can estimate communication parameters having only the bit rate as a priori knowledge. The same work investigates the possibility of man-in-the-middle attacks on FlexRay nodes by interposing a malicious device between the network and the target node. Although efficient, this type of attack requires more complex equipments and physical access that is not always available. Our attacks are simply implemented at the software layer of automotive-grade controllers.

1.2 Paper organization.

Section 2 introduces the theoretical concepts behind the FlexRay protocol focusing on general aspects of the specification. Particular FlexRay behaviour and features enabling attacks are presented and discussed in section 3 which is dedicated to the protocol security analysis for a better understanding of the attacks they make possible. The attacks presented in section 3 are put into practice and analysed in terms of feasibility in the experimental section 4. Finally, section 5 presents the conclusions of our work.

2 FlexRay protocol fundamentals

The FlexRay protocol was developed by the FlexRay consortium founded by several major automotive OEMs (Original Equipment Manufacturers) later joined by electronics manufacturers. The latest specification version published by the consortium in 2010, before being transferred to an ISO standard, is 3.0.1. and includes information on the physical layer [1] and the actual protocol specification [2]. Currently the FlexRay protocol specification is provided as the multi-part standard ISO 17458. Part 1 [8] of the standard covers general aspects of the protocol like use case scenarios and terminology, while parts 2 [9] and 4 [10] cover the FlexRay data link layer and physical layer respectively.

Given its intended use for high performance and safety-critical applications FlexRay was designed to provide deterministic, fault tolerant and high-speed data transmission. Deterministic communication is assured by employing a time-triggered communication model while fault tolerance is provided through communication channel redundancy as two independent channels are supported. Each of the two channels available on FlexRay capable nodes can achieve bit rates of up to 10 Mbit/s. When communication redundancy is not required, the two channels can be used for simultaneous data transmission increasing the bit rate up to 20 Mbit/s.

2.1 Communication architecture

By specification FlexRay is not limited to a certain network topology. It can be used to implement point-to-point communication as well as bus, passive or active star and even hybrid network topologies. The active star topology is implemented by the use of an active star coupler which improves protection against error propagation, allowing the disconnection of faulty nodes.

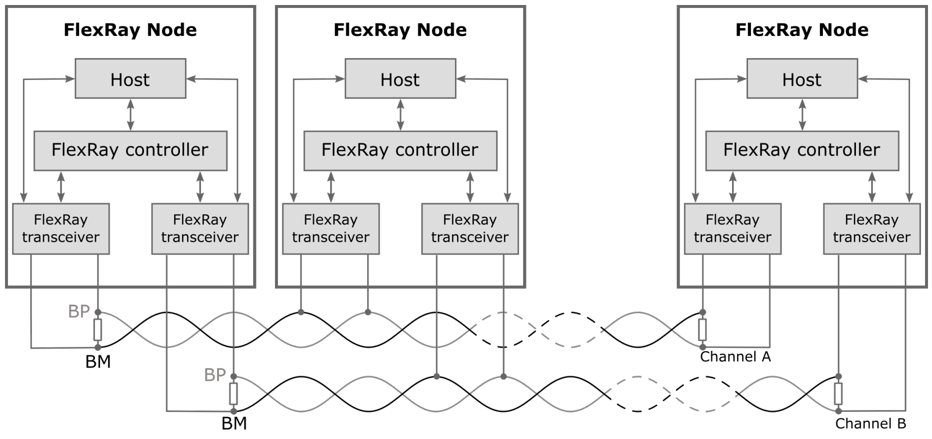

Fig. 1. FlexRay nodes connected in a bus topology

As illustrated in Figure 1, each FlexRay node consists of a transceiver, controller and host. The transceiver or bus driver is responsible with the physical layer signaling part of the FlexRay communication stack while the controller implements the protocol logic corresponding to the data link layer. All higher level logic has to be implemented in the host application.

2.2 Physical signaling

FlexRay uses a two line differential interface to implement the physical signaling. The FlexRay transmission medium consist of a pair of twisted lines denoted as *BP* (bus plus) and *BM* (bus minus) which must be fitted with proper termination at its ends. The FlexRay physical signaling is based on four line levels, as illustrated in Figure 2, two recessive levels to signal low power or idle mode and two dominant levels representing the two logical line levels: *Data_0* (logical zero) and *Data_1* (logical one).

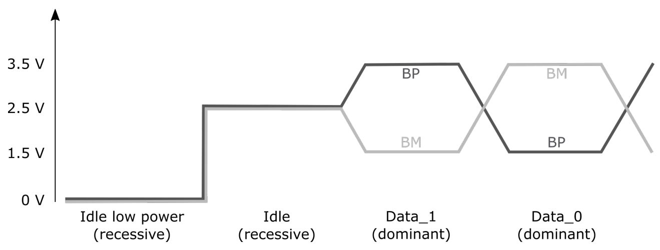

Fig. 2. FlexRay physical bus levels

2.3 Channel access

Access to the communication channel is granted based on the TDMA (Timed Division Multiple Access) method. This implies that the communication takes place according to a predefined schedule which is executed periodically as the FlexRay communication cycle. The communication schedule is defined at the network design time and must be followed by all nodes.

The structure of the FlexRay communication cycle allows the transmission of messages that have to be sent periodically as well as sending sporadic messages. These segments of the FlexRay communication cycle are depicted in Figure 3. The communication cycle consists of at least two segments, the static segment and the network idle time (NIT). The static segment is dedicated to deterministic message transmission, while the NIT is a time interval with no message transmission required for clock synchronization. Two other optional segments can be also present in the communication cycle, the dynamic segment and the symbol window. The dynamic segment is used for event-triggered message transmission and must always follow the static segment. The symbol window is used to signal specific FlexRay activities such as communication wake-up or a node joining the communication by the use of special bit sequences called symbols.

The bulk of the FlexRay communication occurs during the static and dynamic (if present) segments. The static segment consists of a number of static slots equal in length that can be assigned to periodic messages. The static segment can hold at most 1023 static slots and must be configured to accommodate

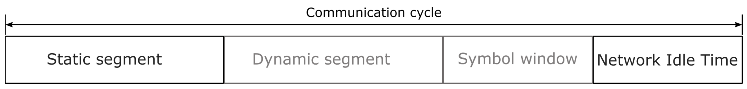

Fig. 3. Components of the FlexRay communication cycle

at least 2 slots. The messages that must be transmitted in the dynamic segment also have to be defined in the communication schedule. They will only be transmitted if the event triggering their transmission occurs in this segment. The dynamic segment always has the same length. Therefore, if the allocated time interval is not enough to accommodate all dynamic messages marked for sending in the current cycle then the unsent messages will be deferred to the next cycle. Transmission priority in the dynamic segment is based on the message schedule such that messages with lower ID values will be sent first.

2.4 Frame format

FlexRay data transmission is done using dedicated frames which, as depicted in Figure 4, are composed of three main segments: header, payload and trailer. The frame header consists of 5 bytes carrying a set of indicator bits, the frame identifier (ID), payload length, header CRC and cycle count. The five indicator bits are to be interpreted as follows: a) *Reserved bit* - reserved for future protocol use and should be always set to 0 when transmitting, b) *Payload preamble indicator* - when set to one it indicates that the payload segment contains specific optional information depending on the segment used to transmit the frame, c) *Null frame indicator* - when set to zero indicates that the payload segment contains no valid data, d)*Sync frame indicator* - when set to one it indicates that all receiving nodes should use the frame for synchronization purposes, and e)*Startup frame indicator* - when set to one it indicates that the current frame is a startup frame which is used by the startup mechanism.

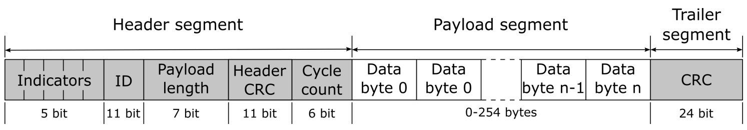

Fig. 4. The FlexRay frame format

The ID defines the slot in which the frame should be sent, hence it will be sent at most one time during a communication cycle. The payload length field indicates the size of the payload as a multiple of 16 bit words. The header CRC value is computed over the last two indicator bits, frame ID and payload length fields. The cycle count represents the current value of the cycle count from the sender's perspective.

The payload segment contains the actual data transmitted and can hold between 0 and 254 bytes. It will always contain an even number of bytes as a consequence of the payload length indicating its size as a multiple of two byte words. The trailer segment contains the CRC computed over the entire frame.

2.5 Error handling

The FlexRay error handling mechanisms uses three states to achieve error confinement: *normal active*, *normal passive* and *halt*. Normal active is the normal operation state in which it is assumed that the node can perform all its activities. In the normal passive state the node is not allowed to transmit as it is assumed that collisions may occur on transmission attempts due to existing synchronization errors. The halt state is entered when detected errors are considered severe enough that proper protocol operation can only be re-established by reinitializing the node.

There are two main mechanisms for handling errors. The first is used for significant errors and causes transitions directly into the halt state. The second mechanism employs a degradation model based on switching between the three operation states to avoid immediate transitions to the halt state in case of transient errors. The degradation model handles synchronization related errors. Direct transitions to the halt state are made on product-specific errors, fatal errors during frame and symbol processing or direct host command.

3 Specification analysis

We will next present elements of the FlexRay protocol specification that can be exploited to mount various attacks. Each vulnerability is introduced considering a passive network topology. We also discuss the identified vulnerabilities from the perspective of an active star network topology in a dedicated section.

3.1 Attacker model

The vulnerabilities discussed in this section are considered as seen from the perspective of an attacker with the intent of disrupting or faking FlexRay communication. This attacker has reasonable to good knowledge of the FlexRay protocol or the ability to gain the knowledge by studying the publicly available specification. The attacker has the ability to compromise the firmware of an in-vehicle FlexRay network node either by updating the firmware through available channels (e.g. OBD port or OTA mechanisms) or by providing an already compromised node to be fitted as a replacement or after-market component. We also assume that the attacker is capable of controlling HW components of the compromised node through SW but not able to effect any HW changes. The attacker can not make any changes to the target FlexRay network topology such as interposing a malicious node between an existing one and the rest of the network.

3.2 Physical layer

The functionality of the FlexRay protocol primarily relies on the ability of generating the basic physical line signals corresponding to the logical bus states Idle, Data_0 and Data_1, depicted in Figure 2. The FlexRay electrical physical layer specification [1, 10] defines differential voltage thresholds for the detection of the logical bus states. Since FlexRay has no integrated means of resolving collisions, during normal operation it is only allowed for one node to generate a dominant signal (i.e. Data_0 or Data_1) at a moment in time while all other nodes have to generate an Idle (recessive) signal. In case of colliding dominant signals the detected line level cannot be predicted as the resulting line voltage level will depend on the voltage levels of the generated dominant signals. Therefore, generating collisions will lead to perturbations of the FlexRay communication. This can be easily achieved on a FlexRay node by directly interacting with the FlexRay transceiver which is responsible of generating physical bus levels based on logical input data coming from the host microcontroller. Normally the transceiver input lines are controlled through the communication controller but this can also be done by using the microcontroller's I/O ports. This approach could be used to implement several variants of DoS attacks as presented in what follows depending on the intended effect.

Full DoS attack. The FlexRay communication could be completely blocked by generating a continuous dominant level on the bus. For this, the attacker should assure corresponding high, or low constant levels on the transceiver's transmit data (TxD) pin and enable the transmitter circuit by setting the transmit enable (TxEN) pin to low. Due to protection circuitry in the transceiver it may be required to periodically assure short high level pulses on the TxEN.

While, the effect of this attack is to prevent any FlexRay communication, the reasons for achieving this depend on the moment of the attack launch. If the attack is initiated at system startup before initiating the FlexRay communication, the nodes will not be able to successfully perform wakeup and startup activities preventing any data transmission. Starting the attack after the communication has been successfully initiated will lead to all nodes being unable to interpret any sent messages. As a result the resynchronization tasks performed on each node will fail for all subsequent communication cycles. After a configurable number of failed resynchronization attempts all nodes will enter the normal passive followed by the halt state.

Targeted DoS attack. The attacker may intend to only prevent a certain node from sending messages or just certain messages from being sent. To achieve this the attacker node needs to generate a continuous dominant level during the transmission of the target message that leads to failure on the receiver side in correctly interpreting the message. Any failure in verifying correct frame encoding or frame content integrity (i.e. CRC verification) will result in the frame being ignored by the receiver. Knowledge about the network and schedule configuration is needed in order to mount this type of attack but this might be automatically achieved by analysing network traffic as demonstrated in [3, 6, 7].

While static frames have a clearly defined slot allocated for their transmission, the transmission slot of dynamic frames within the dynamic segment cannot be predicted. Therefore, employing this attack approach for preventing the transmission of certain dynamic frames might prove to be more difficult as it would require additional real-time traffic analysis.

3.3 Data-link layer

The FlexRay communication controller implements the data link layer part of the protocol. The correct functioning of the FlexRay communication relies on using the communication controller along with a proper configuration of FlexRay protocol parameters and common knowledge of the communication cycle on each network node. However, this configurability makes it possible to misuse protocol parameters and communication cycle settings directly from the application layer. We will discuss specific elements of the FlexRay protocol and the attacks they can enable in the following sections.

Full DoS attack. Improper configuration of the FlexRay communication cycle on one network node can generate protocol errors on all other nodes and can potentially lead to a suspended communication if synchronization is lost. This issue can be exploited to mount a full DoS attack on the network by purposely affecting node synchronization.

A way to achieve this is for the attacker node to send messages in all slots of the communication cycle causing collisions with messages from any of the other nodes. This will result in a loss of synchronization due to the inability of legit nodes to correctly receive synchronization frames. The loss of synchronization occurs after a configurable number of resynchronization attempts. In this version of the attack it is not necessary for the attacker to know detailed information about the communication schedule of the target network. It is enough to know the communication cycle length and bit rate on which to build a setup that will allow integration in the existing communication to enable sending messages in all slots so that collisions are guaranteed.

A more targeted approach would be to only generate collisions on synchronization frames. When defining the communication schedule it is specified which frames should be used for synchronization. As presented in the FlexRay frame description, these frames can be identified by the Sync frame indicator bit in the frame header. This makes it easy for the attacker to identify the slots used for synchronization. It would be enough for the attacker to generate collisions by sending messages in the slots allocated for these frames to force loss of synchronization. Based on its implementation methodology this attack could be considered a special case of the targeted DoS attack which we detail next. Here the target messages are all synchronization frames but the intended end result is complete communication halt.

Targeted DoS attack. Similar to the case of transceiver-based attacks, mounting targeted DoS attacks using the FlexRay communication controller aims to prevent certain network traffic for being correctly received. This can be easily achieved by adding target messages to the attacker node's own communication schedule setup and transmitting them in the appropriate slots. This will cause collisions on the target messages and decoding errors on the receivers side leading to the message being rejected. FlexRay does not provide any means of signaling failed transmissions to the sender node, these are only reported to the hosts of the receiver nodes.

While mounting this attack on static messages is straight forward, targeting dynamic frames will require sending the target frame in each communication cycle since the attacker has no knowledge on the occurrence of message transmission trigger event on the legitimate message sender side. This results in a combined DoS and spoof attack efficient in feeding receiving nodes false data while preventing the legit node from intervening.

Message spoofing. The methodology behind message spoofing is similar to the one employed for the targeted DoS attack. The attacker node has to define the target messages in its own communication schedule. To assure its correct transmission the spoofed message has to be sent inside communication cycles which do not contain the message transmitted by the legit node. To increase attack efficiency the legit message transmission could be prevented by generating a collision.

Messages sent in the static segment are cyclic messages which are sent with a certain periodicity that is a multiple of the communication cycle period. According to the FlexRay specification, a node should always transmit a frame in a static slot assigned to it regardless of data availability for the particular communication cycle. The node should transmit a null frame in the static slots for which there is no data ready to send. This behaviour makes it impossible to spoof messages in a static slot as long as it is assigned to an active legit node since any attempt to do so will result in a collision.

Spoofing messages assigned to the dynamic segment is possible since their corresponding slots will only be occupied if there is data to transmit. Spoofing dynamic frames might result in collisions as it is impossible to know when the legit node will send a message in the same slot. Similar to the case of targeted DoS attacks this effect works to the advantage of the attacker rendering the legit node unable to send correct data while the attacker transmits faked frames.

3.4 Feasibility of attacks in networks based on active star

Given their nature, networks built on an active star topology could be used to prevent attacks.

We discuss the feasibility of the previously proposed attacks on active star-based FlexRay network topologies by strictly referring to the effect of the attacks on the nodes connected to the attacker node through the active star coupler. It is obvious that the behaviour of the proposed attacks on the direct link between the active star coupler and the attacker node will be as already described.

According to the FlexRay physical layer specification [1, 10] the star coupler can set a branch (i.e. physical link) in one of two states for preventing the propagation of communication misbehaviour from it to other branches, i.e., *Branch_FailSilent* and *Branch_Disabled. The Branch_FailSilent* state is entered as a result of a bus error detection which is specific to particular transceiver implementations. A transition to the *Branch_Disabled* state is done on host control and depends on specific implementations of the host. Therefore, the star coupler could implement the ability to detect and block any misbehaving traffic from reaching other network branches. The FlexRay specification does not include specific requirements or recommendations for misbehaviour detection and handling for the active star coupler besides switching the transmitter state to off after exceeding the maximum allowed length of transmitter activation. This protection mechanism can be overcome by periodically toggling the attacker transmitter state as discussed in the section dedicated to physical layer based attacks. In this context, it may be possible to implement all previously presented attacks on active star based FlexRay networks if proper user-defined protection mechanisms are not implemented.

Compromising the active coupler node would make it possible to spoof and DoS any traffic (including static segment frames) from directly connected nodes since all message routing would be under the control of the attacker. Without introducing additional security mechanisms it would be virtually impossible for the other nodes to detect any attack launched by the active star coupler node.

4 Experimental analysis

4.1 Experimental setup

To evaluate the feasibility of the proposed attacks we built a FlexRay network containing 4 nodes connected according to a bus topology. One of the 4 nodes is always used as the attacker node. Each node of our test network is built on an EVB9S12XF512E development board equipped with a Freescale (NXP) S12XF512 microcontroller two TJA1080A FlexRay transceivers (dedicated to the two FlexRay channels). The employed microcontrollers are equipped with 32Kbytes of RAM and 512Kbytes of Flash and can operate at a top frequency of 50MHz. They are intended for automotive applications that require low to medium performance. Figure 5 illustrates our experimental setup with the cluster of 4 FlexRay nodes and a PicoScope used to analyse network traffic. In particular, the FlexRay protocol decoder provided by the PicoScope application was used to identify malformed frames resulted from the tested attacks.

For the legit nodes we defined a communication schedule that includes transmissions both in the static and dynamic segment. Each communication cycle uses the same slot assignment for both the static and dynamic segments as shown in Table 2 where A_i, B_i, and C_i are messages sent by nodes A, B and C

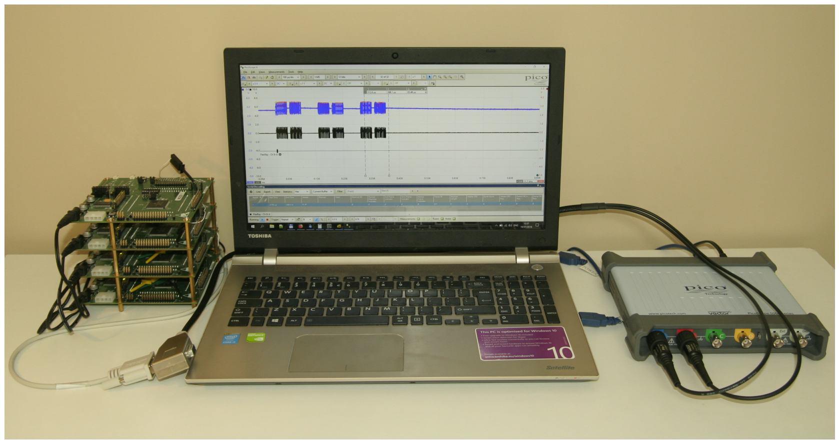

Fig. 5. Experimental FlexRay network setup used for testing proposed attacks

respectively. To simplify the attack implementations we considered that, where required, knowledge about communication parameters are already at hand. This is a realistic assumption as past research has shown it is possible to extract FlexRay communication parameters by listening to existing traffic [3, 6].

Table 2. Slot assignment within the test communication cycle.

	Static segment															**Dynamic segment**							
Slot	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23
Message	A_1	A_2		B_1	B_2		A_3	B_3	C_1	C_2			C_3							C_4		A_4	B_4

4.2 Full DoS attack evaluation

Transceiver-based full DoS. We tested this attack by using it to prevent the start of FlexRay communication as well as force the end of an already established communication. In the first case, legit nodes attempted to establish communication in the presence of the attacker generated signal but are not able to do so as neither node is able to receive correct data. The second attack scenario resulted in the nodes attempting to achieve resynchronization for a configurable number of consecutive odd cycles before entering the passive state and stop transmitting. Figure 6 a) exemplifies the result of the attack on already established communication by generating a permanent Data_1 level on the communication channel. The number of resynchronization attempts parameter was set to 5 in our case, hence the 10 additional cycles following the attack start. The nodes eventually enter the halt state which can only be exited if the host restarts the FlexRay controller. We investigated both generating permanent Data_0 and permanent Data_1 for the attack and achieved the same end effect in both cases.

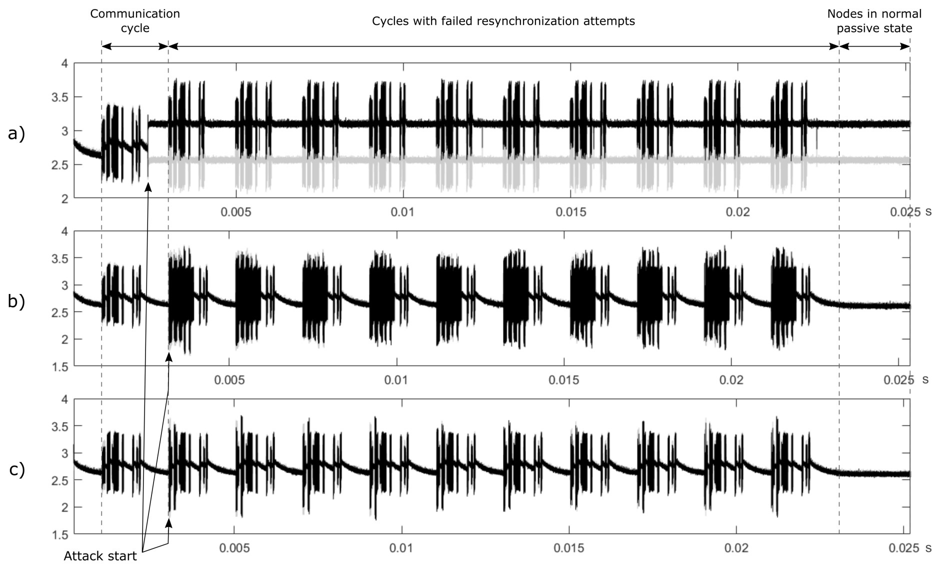

Fig. 6. Full DoS attacks: a) transceiver-based version, b) controller-based version targeting all static frames, c) controller-based version targeting only synchronization frames

Controller-based full DoS. For the implementation of this type of attack and the subsequent controller-based attacks we configured the FlexRay controller on the attacker side with the same communication parameters employed by the other nodes. We tested several variants of the full DoS attack first by creating collisions on the entire communication schedule, then only on the static segment frames and finally by targeting only the synchronization frames. In all cases, the implementation is straight forward as it only requires including messages targeted for collision in the communication schedule of the attacker and making sure they are sent in the defined slot. The result in all cases was, as expected, the termination of FlexRay communication once all nodes lost synchronization and entered the normal passive state in which a node is not allowed to transmit. From this state all nodes will then transition to the halt state since in this case there is no method of regaining synchronization in the absence of traffic containing synchronization frames. Figure 6 b) and c) illustrate the effect of the attack on all static segment frames and all synchronization frames respectively. The attack should persist long enough for all the nodes to enter the halt state. This is achieved by setting the maximum allowed cycles without clock correction on the attacker side to the maximum value used within the network or maximum allowed value according to the specification which is 15.

Collisions occur when sending frames in a slot allocated to another node regardless of the frame content. If the attack frame is other than the legit frame the result of distinct dominant levels colliding is undefined and cannot be interpreted. Even if the attack frame is identical to the legit one the resulting physical levels are still in violation of protocol specification making the frame non-decodable.

Comparing attack variants. As illustrated in Figure 6 all variants of the full DoS attack lead to the same end result, i.e. the transitioning of all legit nodes in the halt state and the end of message transmission. In terms of attack complexity, the transceiver-based attack requires less knowledge on the FlexRay protocol and communication schedule configuration.

4.3 Targeted DoS attack evaluation

Transceiver-based targeted DoS. We implemented this type of attack by using the timer module to identify the start of a communication cycle by detecting the appearance of the NIT and then measure the offset from the cycle start to the target frame. Once the target frame slot is detected, the attacker node starts generating the continuous dominant bus level ending the attack no later than the frame slot end. The result, as shown in Figure 7 a), is that collisions will be generated on transmitted target frames making the receiver side unable to correctly decode the frame. All other traffic passes unaffected. Both targeted nodes and other communication participants continue fulfilling the communication cycle as the attack does not generate any transitions in the error confinement mechanism. Receiver nodes could detect the missing frame in the expected slot and implement a special handling for this case at the application layer.

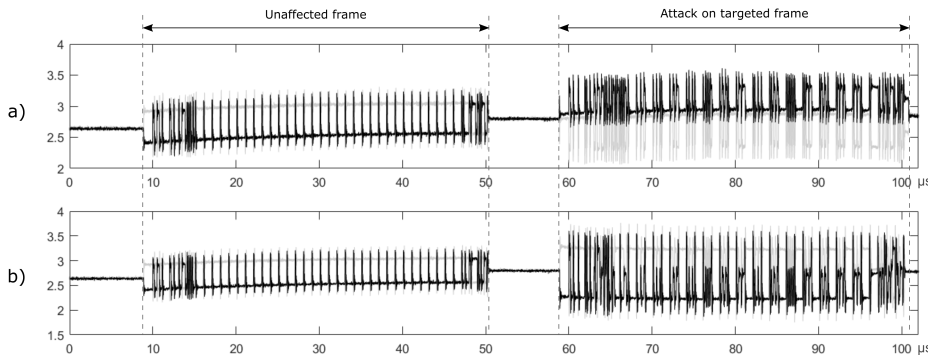

Fig. 7. Targeted DoS attacks: a) transceiver-based version, b) controller-based version

Controller-based targeted DoS. The approach for mounting the targeted DoS attack using the FlexRay controller was similar to the case of the full DoS attack. Target messages were added in the attacker's communication schedule end transmitted in the corresponding slot. As a result, like in the previous attack version, nodes are unable to receive targeted frames but communication continues as long as unaffected synchronization frames are still sent to maintain synchronization. This effect of this attack on the physical line levels is depicted in Figure 7 b).

Comparing attack variants. While the end result of the targeted DoS attack is the same in both attack variants, the effort of implementing the transceiver-based version is higher in contrast with the controller-based approach which only requires adding the target slot in the attacker's communication schedule.

4.4 Message spoofing attack evaluation

We checked the behaviour of the FlexRay controller on our S12XF platform when configuring message transmission periods for the static slots to be greater than the communication cycle period. As specified, all unused occurrences of allocated slots were filled by the communication controller with a null frame transmission making it impossible to mount a spoofing attack on the static segment messages.

On the dynamic segment side we were able to implement message spoofing by assuring periodic transmission of the injected frame. As expected, some of the spoof transmissions collided with the legit on-event frame stopping it from being correctly received.

5 Conclusions

Our work brings more experimental insights but also theoretical discussions on the feasibility of attacks on FlexRay networks. The DoS and spoofing attacks that we experiment with are relevant for FlexRay due to the safety-critical nature of in-vehicle communications. In terms of countermeasures, the possible approaches depend on the employed network topology. In case of passive topologies (i.e. point-to-point, bus or passive star), message spoofing attacks can be prevented by adding proper cryptographic authentication mechanisms while DoS remains a more demanding issue. Active star topologies with specific message filtering mechanisms implemented in the active star couplers can circumvent both DoS and spoofing attacks. While using an active star topology looks like a good approach from the security point of view it may not suit all applications. In such cases a hybrid topology approach could be used to separate parts of the FlexRay network through an active star coupler. It is good news that FlexRay supports such topologies and given the safety-critical nature of in-vehicle devices, more efforts in this direction may be desirable.

Acknowledgements

This work was supported by a grant of the Romanian Ministry of Research and Innovation, CNCS - UEFISCDI, project number PN-III-P1-1.1-PD-2016-1198, within PNCDI III.

References

1. FlexRay Communications System - Electrical Physical Layer Specification, Version 3.0.1. Standard, FlexRay Consortium (2010)
2. FlexRay Communications System - Protocol Specification, Version 3.0.1. Standard, FlexRay Consortium (2010)
3. Armengaud, E., Steininger, A., Horauer, M.: Automatic Parameter Identification in FlexRay based Automotive Communication Networks. In: Emerging Technologies and Factory Automation, 2006. ETFA'06. IEEE Conf. on. pp. 897–904. IEEE (2006)
4. Checkoway, S., McCoy, D., Kantor, B., et al.: Comprehensive Experimental Analyses of Automotive Attack Surfaces. In: USENIX Security Symposium (2011)
5. Cho, K.T., Shin, K.G.: Error handling of in-vehicle networks makes them vulnerable. In: Proceedings of the 2016 ACM SIGSAC Conference on Computer and Communications Security. pp. 1044–1055. ACM (2016)
6. Heinz, M., Höss, V., Müller-Glaser, K.D.: Physical layer extraction of FlexRay configuration parameters. In: Rapid System Prototyping, 2009. RSP'09. IEEE/IFIP International Symposium on. pp. 173–180. IEEE (2009)
7. Huse, M.I.: FlexRay Analysis, Configuration Parameter Estimation, and Adversaries. Master's thesis, NTNU (2017)
8. ISO: 17458-1, Road vehicles – FlexRay communications system – Part 1: General information and use case definition. Standard, International Organization for Standardization (2013)
9. ISO: 17458-2, Road vehicles – FlexRay communications system – Part 2: Data link layer specification. Standard, International Organization for Standardization (2013)
10. ISO: 17458-4, Road vehicles – FlexRay communications system – Part 4: Electrical physical layer specification. Standard, International Organization for Standardization (2013)
11. Koscher, K., Czeskis, A., Roesner, F., et al.: Experimental security analysis of a modern automobile. In: Security and Privacy (SP), 2010 IEEE Symposium on. pp. 447–462. IEEE (2010)
12. Miller, C., Valasek, C.: Adventures in automotive networks and control units. DEF CON **21**, 260–264 (2013)
13. Murvay, P.S., Groza, B.: DoS Attacks on Controller Area Networks by Fault Injections from the Software Layer. In: Proc. of the 12th Intl. Conf. on Availability, Reliability and Security (ARES'17), 3rd Intl. Workshop on Secure Software Engineering (2017)
14. Nilsson, D.K., Larson, U.E., Picasso, F., Jonsson, E.: A first simulation of attacks in the automotive network communications protocol flexray. In: Proc. of the Intl. Workshop on Computational Intelligence in Security for Information Systems CISIS'08. pp. 84–91. Springer (2009)
15. Palanca, A., Evenchick, E., Maggi, F., Zanero, S.: A stealth, selective, link-layer denial-of-service attack against automotive networks. In: International Conference on Detection of Intrusions and Malware, and Vulnerability Assessment. pp. 185–206. Springer (2017)
16. Wolf, M., Weimerskirch, A., Paar, C.: Security in automotive bus systems. In: Workshop on Embedded Security in Cars (2004)