Accommodating Time-Triggered Authentication to FlexRay Demands

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

Lucian Popa
Politehnica University of Timisoara
Timisoara, Romania
lucian.popa@aut.upt.ro

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

ABSTRACT

Research efforts related to in-vehicle communication security were largely focused on the Controller Area Network (CAN) protocol. While CAN is still the most widely used protocol for building in-vehicle networks, many safety critical functionalities are based on other communication protocols such as FlexRay or Ethernet which constantly expand their use inside vehicles. In this paper we address the problem of authenticating transmissions in FlexRay networks. We approach this task by adapting an authentication protocol to the time-triggered nature of FlexRay communication while also accounting for non-deterministic transmissions that may occur in the FlexRay dynamic segment. We illustrate the effects of introducing authentication on keeping strict message deadlines by evaluating our proposal based on a real-life scenario from a major vehicle manufacturer.

CCS CONCEPTS

• **Security and privacy** → **Security protocols**; • **Computer systems organization** → *Embedded systems*;

KEYWORDS

automotive, security, FlexRay, authentication

ACM Reference Format:

Pal-Stefan Murvay, Lucian Popa, and Bogdan Groza. 2019. Accommodating Time-Triggered Authentication to FlexRay Demands. In *Central European Cybersecurity Conference (CECC 2019), November 14–15, 2019, Munich, Germany*. ACM, New York, NY, USA, 6 pages. https://doi.org/10.1145/3360664.3360666

1 INTRODUCTION AND MOTIVATION

Considerable effort was dedicated, both by the academic community and industrial sector, into designing security mechanisms for in-vehicle communication as a result of the many reported attacks [4, 10] on these automotive networks. However, existing works in this area are largely focused on the Controller Area Network (CAN) protocol which was targeted in most of the attacks and which is still the most widely used protocol in the automotive industry. The features of CAN are suitable for many current vehicular applications. But there are some functional domains, such as power train, chassis and advanced driver assistance systems, which provide safety critical functions that have higher requirements in terms of bandwidth and reliability compared to CAN's capabilities. Other technologies such as FlexRay or automotive Ethernet were introduced to provide the required communication capabilities for these functions.

In particular, FlexRay provides higher bandwidth and increased reliability for time-critical applications. Like other wired in-vehicle communication protocols, FlexRay also lacks in terms of security mechanisms. The earliest mentions of exploitable FlexRay vulnerabilites date from 2004 [20]. Since then, other works have illustrated attacks on FlexRay communication using either simulations [12] or actual experimental implementations [11] of FlexRay networks. Since FlexRay is almost entirely used in safety critical applications, attacks on FlexRay communication will have serious implications on human safety (i.e. driver, passenger or other traffic participants).

In this paper we focus on the security of FlexRay by addressing the topic of authenticated communication. We consider an authentication protocol based on symmetric primitives and one-way key-chains as main building blocks. We take advantage of the time-triggered nature of FlexRay communication by utilizing keys associated to time intervals. Key distribution and different approaches for leveraging communication and computational overheads are discussed. We also illustrate the impact of the authentication mechanisms on the ability to keep deadlines by experimenting on a setup built after the specification of a real-life scenario. We continue by presenting related work and FlexRay protocol basics in sections 2 and 3 respectively. Section 4 presents our proposal for approaching authentication in FlexRay networks and discusses the security of the proposed scheme. Finally, in section 5, we present experimental results before giving our conclusion.

2 RELATED WORK

There are several lines of work that tackled the issue of authentication in FlexRay or other time-triggered networks. Szilagyi et al. [16] discusses the use of Message Authentication Codes (MACs) and keys shared between node pairs for authentication in time-triggered networks. The main disadvantage of their approach is that it introduces a considerable communication overhead by transmitting one MAC for each receiver node. Another work on time-triggered systems [19] uses a variant of the TESLA protocol [13] with hash key chains and timed release of keys. Computational and memory overheads are presented based on several ARM platforms, however, the authors indicate that the computational overhead might be too large for real-life applications requiring the use of stronger processors or cryptographic hardware.

CECC 2019, November 14–15, 2019, Munich, Germany

ACM ISBN 978-1-4503-7296-1/19/11...$15.00
https://doi.org/10.1145/3360664.3360666

In [17] a range of broadcast authentication protocols proposed for CAN were mapped and evaluated on a FlexRay based simulated network. The paper presents computational and communication overheads for each case. On the downside, no adaptions of the protocols was done to better fit the FlexRay communication model.

Püllen et al. investigate in [15] the use MACs and hash key-chains to achieve frame authentication. They discuss different strategies for MAC transmission and sharing hash chain seeds between recipient nodes while associating keys to time slots. However, there are less discussions on the practical performance of the proposed schemes. In fact, a common shortcoming of all the previous works is the lack of an evaluation for the authentication overhead on schedule deadlines based on real-life scenarios.

Other lines of work addressed the problem of scheduling of the FlexRay communication. The scheduling problem for a FlexRay network using the TESLA authentication protocol is discussed in [7]. Under the assumption that some network nodes can be equipped with Hardware Security Modules (HSMs) to reduce the computational overhead, the authors determine the minimum number of HSMs required depending on the message deadlines and number of tasks running in the network. Since the main focus of the paper was on formulating and solving the scheduling problem, details on efficient deployment of TESLA for FlexRay are limited.

3 THE FLEXRAY PROTOCOL

The FlexRay protocol was developed by a consortium of representative companies from the automotive sector. A new protocol was needed since CAN, the de facto standard for automotive communication, proved its limitations in fulfilling the requirements of modern high-performance and safety-critical applications. To comply with timing requirements of safety-critical functions, FlexRay was designed to offer deterministic transmissions following a time-triggered communication model. Moreover, FlexRay provides fault-tolerance by means of its fault detection and confinement mechanisms as well as through channel redundancy. Each FlexRay node comes with two channels each capable of bit rates of up to 10Mbit/s. If redundancy is not required, simultaneous transmission on both channels is possible allowing overall bit rates of up to 20Mbit/s.

At the physical layer [1] FlexRay is implemented as a two-wire differential line. Various topologies can be employed to build FlexRay-based networks ranging from bus to active or passive star. Hybrid topologies are also possible as long as propagation delays between network nodes that are farthest apart from one another comply with the FlexRay specification.

The largest part of the FlexRay protocol is defined in the Data-Link layer specification [2]. This document describes the communication cycle and frame format which we detail next.

3.1 Communication cycle

FlexRay uses a TDMA (Timed Division Multiple Access) mechanism to establish node access to the communication medium. Accordingly, the communication follows a cyclically executed predefined schedule based on the FlexRay communication cycle. The communication schedule is defined off-line during network design and has to be adopted by all nodes for a collision-free communication.

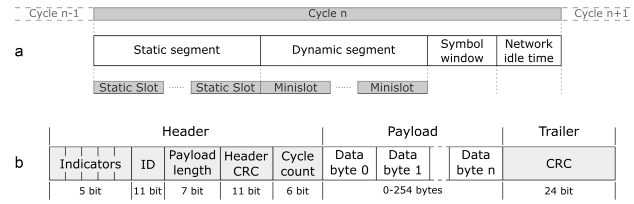

Figure 1: The FlexRay communication cycle (a) and frame structure

At the macro-level, a FlexRay communication cycle, depicted in Figure 1 (a), consists of 4 sections: static segment, dynamic segment, symbol window and network idle time (NIT). Out of these, only the static segment and NIT are mandatory. The predefined size of each of these segments is always the same in each communication cycle.

The main body of the traffic, following a time-triggered model, takes place in the static segment which is divided in a number of static slots each of which can accommodate a single frame transmission. Static slots are equal in size, therefore, all frames transmitted in these slots have the same payload length. Static slots are uniquely assigned to nodes according to the communication schedule.

The role of the dynamic segment is to provide support for non-deterministic (i.e. event-triggered) transmissions. This is achieved by dividing the segment in a fixed number of so-called minislots. Minislots are then allocated, in a priority based manner, upon request for the transmission of dynamic messages. Thus, the dynamic frame with the lowest value ID (highest priority) will be expected to start in the first minislot. If this happens, then the number of minislots required to transmit the whole frame is allocated for this purpose provided that sufficient minislots are available. If the frame transmission does not start, the slot is consumed and the following frame ID is expected in the next available minislot. This continues until the number of available minislots is not sufficient to accommodate a frame transmission. This handling enables the transmission of different size frames in the dynamic segment.

The symbol window is used for special transmissions employed mainly in the node wake-up process. At the end of each cycle a NIT segment is required to allow nodes to perform synchronization and communication management operations.

3.2 Frame format

The format of the FlexRay frame, illustrated in Figure 1 (b) , was designed to assure integrity and increased data throughput. The logical frame structure is the same for both static and dynamic segment transmissions. It features an 11 bit CRC dedicated to the frame header and an additional 24-bit CRC for the entire frame. The header indicates the type of frame, the ID (which corresponds to the slot dedicated for the frame transmission), the payload length and the current cycle count. The data field can hold up to 254 bytes of actual data. The payload size is always a multiple of 2 bytes as a consequence of the design decision of denoting payload lengths of up to 254 bytes using a 7 bit length field.

Additional bit fields are added at the start and end of frame as well as before each byte for additional support for coping with synchronization related problems.

Table 1: FlexRay communication parameters of three industrial use cases from major automotive manufacturers

System	Node count	Bit rate	Cycle duration	Static segment				Dynamic segment	
				Duration	Slots	Frame payload	Message cycle	Duration	Frame Payload
BMW 7 series [3]	15	10Mbit/s	5ms	3ms	91	16 bytes	2.5, 5, 10, 20, 40 ms	1.99ms	2-254 bytes
Unkn. X-by-Wire [21]	10	10Mbit/s	1/8ms	0.77/7.77ms*	22/222	26 bytes*	1, 8ms	0.23ms*	2-254 bytes*
Ford prototype [14]	8	10Mbit/s	5ms	4ms	24	156 bytes*	2.5, 5, 10ms	1ms	2-254 bytes*

* Values not directly extracted from the cited sources but calculated based on other specified parameters

4 AUTHENTICATING FLEXRAY COMMUNICATION

4.1 Challenges

Designing authentication mechanisms for FlexRay networks comes with the specific challenges of time-triggered networks. Strict message arrival deadlines, which are often as low as several milliseconds, must be kept.

To better illustrate the strict timing requirements we list the communication parameters of three industrial FlexRay use cases in Table 1. The first [3] corresponds to a real-life implementation running on BMW 7 series vehicles while the other two [14, 21] come from prototypical implementations of two other major automotive companies. The three use cases utilize cycle times for static segment messages in the order of milliseconds to tens of milliseconds with the lowest cycle time at 1ms.

Upon adding authentication, the time required by a message to be available at its destination consists of the timings required to perform the following operations: generation of the data to be transmitted, authentication information generation, data transmission, authenticity verification. With the authentication mechanism in place, even the shortest message deadline of the use cases in Table 1 (i.e. 1ms) must be still fulfilled to allow the functioning of the underlying functionality. To achieve this, the authentication protocol design must be considered alongside with the employed embedded platforms.

Automotive-grade embedded platforms are limited both in computational power and available memory when compared to a conventional PC or even some modern smart mobile devices. This adds to the challenge of bringing cryptography inside vehicles.

4.2 Protocol design considerations

Given the constraint nature of automotive embedded platforms we take steps towards alleviating these limitations by making several design choices for a FlexRay authentication protocol.

Key distribution. To eliminate additional overhead generated by key distribution operations during system operation we perform these tasks in the off-line state, i.e., when the car is not in use. A central or master node, which can be represented by the active star coupler (in active star topologies) or a gateway node, is in charge of distributing keys to all other network nodes. We assume that the master node is equipped with superior storage capability and cryptographic processing power in comparison to other network nodes. The key distribution process can be performed either at the end of a driving session (i.e., after the engine is stopped and car is locked) or when a new driving session is started (i.e., after the car is unlocked). The master must assure that, before a new driving session starts, all nodes have sufficient keys to last for the longest possible driving session. If, after a short driving session, sufficient keys are still available on all nodes, the distribution process may be skipped. The master node is responsible for evaluating current key needs, generating new keys or key seeds (depending on the requirements of the underlying authentication protocol) and transmitting them to network nodes encrypted with the current valid key in the chain corresponding to the recipient node. We discuss more on this key distribution process in what follows.

A secret master key K_i, shared between each node $i = 1..n-1$ and the master, is generated and pre-programmed at production time on all nodes. Rather than exposing this master key in the key-distribution process, we also use one-way key chains. This makes a potential attack on the master key harder as it is not directly used in the subsequent keying process. The master shares with each of the other network nodes a one way key chain $\mathbb{SK}_i = \{sk_i^j \mid sk_i^j = H(sk_i^{j-1}), j = 1..l\}, i = 1..n-1$. Here H is a hash function, l is the key chain size and n is the number of network nodes including the master. The initialization value sk_i^0 of each chain is computed by means of secure derivation from some seed s distributed by the master to each sender and the master key K_i, i.e., $sk_i^0 = \mathcal{KD}(K_i, s)$. Each node then generates and stores a key chain of predetermined size. For example, considering a maximum driving session of 8 hours, a car that is running three times per day (excepting breaks for fueling) for 15 years, an $l = 16470$ is required. If all the chain keys would be stored, a node will roughly require 258KB of memory space to store a chain of 128 bit keys. By only storing intermediate chain values and regenerating chain portions upon requirement, the required memory can be reduced considerably. If the amount of storage space still does not allow storing a key chain for the entire vehicle lifetime, then chain reinitialization could be done by secure reprogramming at the service during one of the routine maintenance visits.

Adaptable use of cryptographic algorithms. Since message cycles are short, we can use faster algorithms provided that they have sufficient security for the short life-time of the message. For this purpose, in the experimental section we provide performance results for several cryptographic primitives.

For example, we also include MD5 which is known to have collisions, but these do not affect the HMAC and its security level may be sufficient at small cycle times.

Truncated authentication tags. We address the communication overhead issue by using truncated authentication tags. The number of tag bits is adapted to message cycle duration (i.e., the smaller the cycle the smaller the tag) while assuring that the security level provided is sufficient for the target time interval. Authentication tag truncation is a common practice to reduce communication overhead. NIST specifies 64 bits as an acceptable size for a MAC tag [6]. However, in some cases this may introduce greater overhead than might be acceptable, e.g. in the BMW use case, a 64 bit tag will cause a 50% increase in the frame payload. The same NIST document covers the case of even smaller sized authentication tags but does not recommend the use of tags less than 32 bits in size. We tailor authentication tag sizes for each message type based on message cycle duration, key validity duration and acceptable probability of interpreting forged data as authentic. The likelihood of accepting forged data can be calculated using the tag size and number of times the same key is used for MAC generation [6].

Authentication tag transmission. Another step for reducing communication overhead is efficient transmission of authentication information. We propose two approaches: (i) transmit the tag in the same frame as the corresponding data and (ii) aggregate all tags generated by one sender in a separate frame. In the first case immediate authentication is possible if the key is available on the receiver side while in the second a receiver must buffer a message until the tag frame is received. The second approach only improves on communication overheads if the sender node has to transmit sufficient tags to efficiently occupy the payload of a separate frame.

Time synchronization. The authentication mechanisms that we discuss next require that network nodes have a common knowledge of the current time. We rely on communication cycle slots to assure a common time base since the FlexRay protocol would not function unless nodes are synchronized. Therefore, the FlexRay communication controller assures the synchronization process. A small synchronization error (in the order of hundreds of nanoseconds) given by intrinsic network characteristics such as propagation delay (which is under 400ns for the maximum allowed distance between 2 FlexRay nodes). Although, fully relying on FlexRay intrinsic features for synchronization reduces overhead, this introduces a vulnerability to man-in-the-middle attacks which we address in section 4.4.

4.3 Authentication variants

We investigate two approaches for authentication in FlexRay networks. The first is based on the TESLA broadcast authentication protocol, while the second uses shared one-way chains for providing immediate authentication.

4.3.1 Timed release of keys. The principle of achieving authentication by first transmitting the message along with the authentication tag and later disclosing the key is employed by the well known TESLA protocol [13]. For security reasons, a loose time synchronization needs to be established between the sender and receiver nodes in order to check that the keys are indeed released after the arrival of the tag. For continuous authentication of packets, the broadcasting period is divided into time intervals and a different key is assigned to each interval. Additionally, the disclosed keys must be authenticated. This is achieved by using a one way key chain generated by the sender. The sender then sends the last generated value in the chain, through an authenticated channel, to all receivers as the chain commitment. Receivers can authenticate the keys disclosed in a certain time interval by performing a hash over the key and comparing the result with the commitment, i.e., $k_i^j = H(k_i^{j-1}), j = 1..l$. The commitment is updated to the new key k_i^j upon successful verification.

This approach is a good match to the FlexRay scenario. Firstly, FlexRay requires time synchronization to function and assuring synchronization is an inherent responsibility of the communication controller. Secondly, the FlexRay communication is performed in repeating cycles providing the time intervals to which keys are associated. There are, however, two aspects to address for using a TESLA-like approach for FlexRay: transmitting chain commitments and handling delayed authentication. As previously stated, in our approach, the master node handles key distribution. Thus, to bootstrap authentication based on timed release of keys the master node generates a random seed s_j for each sender node j and broadcasts it encrypted with the current key sk_*^j from the key chain shared with node j. Both the sender and the master nodes then generate a hash key chain of predetermined length. The sender nodes will store this chain while the master only retains the last generated chain value from each such chain as the chain's commitment. The commitments are then transmitted as a single message along with authentication data for each receiver node, i.e. $M_{comm} = \{c_1, c_2, ..., c_{n-1}, t_1, t_2, ..., t_{n-1}\}$, where each c_i is the key chain commitment for one of the $n-1$ sender nodes and $t_i = MAC(sk_*^j, c_1||c_2||...||c_{n-1}), i = 1..n-1$ are the MAC tags for the $n-1$ receiver nodes.

The delayed release of keys also delays the authentication and thus adds to the time until the actual data can be used on the receiver side. This has to be factored in designing the communication cycle. We propose transmitting the keys in a static slot following the slot used to transmit the data and authentication tag but within the same communication cycle. This assures a precise time interval for the transmission of the key which is certain to follow the authentication tag transmission interval so that the key cannot be sent before the tag. For the case of frames in the dynamic segment, the same result is obtained if the key is sent in the frame having $ID = ID_{tag} + 1$, where ID_{tag} belongs to the frame holding the corresponding authentication tag. This assures that the key is sent in a separate time interval after the tag and that no other higher priority (lower ID value) frames are transmitted before the key.

Regarding memory requirements, receivers only have to store a key chain commitment for each sender while senders have to store the entire key chain. For long key chains, memory requirements may become a problem, however, different storage strategies exist to prolong the protocol lifetime [5].

4.3.2 A hybrid approach with shared one-way key chains. The main downside of the previous approach is the delayed authentication. By recognizing that certain nodes may have a higher trust level than others, in this approach we utilize a one-way key chain shared between a sender and some of the receivers. During the key distribution process the master generates a random seed for each sender node, encrypts it with the sender key and the keys of the trusted receivers then transmits these values. The seeds are utilized by each node to generate a one-way chain of predefined length. Keys are then used in reverse order of generation to generate and verify MACs, each key being valid for a predefined period of time. This way message authentication can be immediately done by the receivers who have a higher level of trust and are in possession of the key-chain while the rest have to wait for the timed release of the key.

While having the benefit of providing immediate authentication, this approach also requires more storage space for trusted receiver nodes since each receiver node has to store a one way chain for each distinct sender from which it consumes messages. The required storage space can be reduced, at the cost of introducing additional computational and/or communication overheads, by using key chain based protocol life prolonging techniques such as proposed in [8].

4.4 Security analysis

Attacker model. We consider the case of an attacker that is able to infiltrate the FlexRay network by either attaching a node to the network or by compromising an existing node, e.g., by reprogramming. The attacker is not able to retrieve any keys that might be stored on the compromised nodes or obtain master node keys which are used to distribute keys to other nodes. We consider that the attacker is able to listen to all network traffic and inject messages. Note that injecting messages in slots employed by legit nodes that have not been removed from the network will always result in collisions [11] making the actual injection impossible. We also consider the case in which the attacker can act as a man-in-the-middle (MITM). This can be achieved by either compromising an active star coupler or interposing a node between two sections of the network.

Attack resilience. As shown by previous work [11], the success of message injection attacks is first of all conditioned by the absence of the legit sender of the injected messages. If the attacker manages to overcome this, then the proposed authentication mechanisms will prohibit spoofing and replay attacks as the attacker is not able to generate valid authentication tags. However, a man-in-the-middle (MITM) attacker could mount a cleverer attack on the TESLA-based protocol as we discuss next. As a MITM, the attacker is responsible of relaying all communication from the rest of the network to the network segment targeted by the attack. The attacker could then force the communication cycles of the attacked network segment to lag behind the real communication by constantly delaying relay of the synchronization frames coming from legit nodes from the rest of the network. The delayed frames will still be accepted as valid as long as they can be authenticated and the delay is not big enough to completely break synchronization. This leads to the creation of time domain cliques (i.e., groups of nodes in the same network with common time-base within the clique but different compared to others) a known problem in time-triggered networks [9]. If enough delay is introduced, so that the attacker receives the disclosed keys from the legit network before the expected arrival time of the message and authentication tag on the receiver side, it can start injecting its own messages. But such isolation of the receivers in a distinct time-slot will hold only if the rest of the network does not notice missing messages from the attacked segment as such messages will not arrive as expected anymore because of the increasing delays. If this happens, the nodes can signal an error or even stop sending messages for the nodes in the attacked segment. Another approach could be periodically signing message history using short-term public keys like suggested in [18]. This kind of attack will not be possible when using shared key-chains.

5 EXPERIMENTAL EVALUATION

5.1 Experimental setup

For evaluating the overheads introduced by adding FlexRay authentication, we implemented a scaled-down setup configured based on the FlexRay communication parameters employed in BMW 7 series vehicles [3]. We selected this scenario because it represents an actual implementation in existing vehicles in contrast with the other previously mentioned scenarios which only represent prototypical implementations of FlexRay networks.

Scaled network model. Our setup consists in a two node network following the communication parameters as described in the first line of Table 1: 10Mbit/s bit rate, 5ms cycle with 3ms allocated for the 91 slots in the static segment each holding a frame with 16 bytes of payload and a 1.99ms dynamic segment. We focus on the messages with lowest cycle times of 2.5, 5 and 10ms, i.e. messages sent twice per cycle, once per cycle and once every two cycles respectively since longer deadlines would be easier to keep. A second communication cycle is configured based on the initial one to accommodate payloads of 20 bytes in static slots for transmitting a 32 bit authentication tag and the message in a single frame. This adaption resulted in the decrease of the number of static slots from 91 to 81, while keeping all other settings. More slots could be configured in the same static segment time span if reducing slot idle times but this will restrict maximum network length ([3] considers a maximum cable length of 24 meters).

Even with the use of a reduced network model, our performance results still give a realistic depiction on the behavior of a full network implementation. The presence of other nodes will not affect communication during the static segment since each node only transmits in its allocated slot while additional traffic in the dynamic segment can be easily generated by the two existing nodes.

Embedded platforms. Two types of automotive grade microcontrollers were employed in our experimental evaluation: the NXP S12XF512 and the Infineon AURIX TC297. With a top operating frequency of 100MHz, 32kB of RAM and 512kB of Flash, the S12XF512 platform targets applications requiring low to medium performance. It features two 16 bit cores: the main S12 core and the XGATE coprocessor included to reduce the load of the main core when serving interrupts. On the other hand, the TC297 offers higher performance with three 32-bit cores running at up to 300MHz, 728kB of RAM and 8MB of Flash. Both microcontrollers are equipped with a range of communication modules including FlexRay controllers.

5.2 Performance evaluation

We evaluated both the communication and computational overhead considering different approaches for transmitting authentication data and various cyptographic building blocks.

Communication overhead. Based on the implemented setup we evaluated communication related overheads. In the original setup (with 16 bytes payload) it takes 62.15μs for a frame to be transmitted and made available on the receiver side. When using authentication with timed release of keys on this setup, assuming the MAC fits in the same frame as the message, it takes 82.59μs to transmit the two consecutive frames holding the message, authentication tag and disclosed key. Accordingly, in the second setup with 4 additional bytes for the MAC tag, it takes 67.91 and 90.40μs

Table 2: Overheads for computing MACs on 128 bit messages with 128 bit keys on the S12XF512 and TC297 platforms

CPU	HMAC-MD5	HMAC-SHA1	HMAC-SHA256	CMAC-AES128
S12	3.034ms	7.118ms	17.45ms	1.381ms
XGATE	1.328ms	3.240ms	8.702ms	538.2µs
TriCore	22.42µs	31.17µs	59.25µs	41.66µs

to receive one and two consecutive frames respectively. If transmitting the authentication tags as a separate frame the incurred authentication delay will vary for each frame depending on the moment of MAC transmission.

Clearly, from the communication point of view it would be more efficient to transmit the authentication tag in a single frame along with the message and use shared key chains to avoid additional frame transmissions for tags or key disclosure.

Computational overhead. In our computational overhead analysis we consider the use of 128 bit keys for MAC generation in accordance with current key length recommendations for symmetric primitives[1]. In Table 2 we present execution times for different MAC algorithm implementations, from the wolfCrypt library[2], on the two employed platforms accounting for the different processing units available on each. As expected, the 16-bit CPUs of the S12 are clearly outperformed in all cases by the TriCore. Even so, thanks to the XGATE co-processor it would take $\approx$ 1.1ms for S12 nodes to compute two AES based CMACs, one before transmission and one for verification upon arrival on the receiver. This fits within the lowest message cycle time of 2.5ms with almost 1.5ms left for data acquisition and transmission. However, the S12 platform will not be able to cope with lowest message cycle time of 1ms from the presented use cases. Higher performance controllers such as the TC297 or dedicated cryptographic hardware should be selected for complying with very short deadlines.

Combined overhead. For low performance platforms the communication overhead is negligible compared with the computational time. High performance platforms, like the Aurix, provide combined overheads of roughly double the transmission times.

6 CONCLUSIONS

We investigated the adaption of time-triggered authentication to cope with the strict requirements of FlexRay communication. Associating keys from one way chains to time intervals for authentication matches the time-triggered nature of FlexRay. As shown, automotive-grade embedded platforms are suitable for providing authentication at various performance levels matching the timing requirements of different time-critical applications. Careful matching of utilized embedded platforms and underlying cryptographic primitives to expected deadlines will provide reliable and efficient implementations of authenticated FlexRay communication.

Detailed protocol design and in-depth security analysis (i.e. formal analysis) of the mechanisms discussed here is required for real-life deployment. We plan these as future work along with optimized generation and storage of the one-way key-chains to cope with both timing and memory demands.

[1]https://www.keylength.com
[2]https://www.wolfssl.com/products/wolfcrypt-2/

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] 2010. *FlexRay Communications System - Electrical Physical Layer Specification, Version 3.0.1.* Standard. FlexRay Consortium.
[2] 2010. *FlexRay Communications System - Protocol Specification, Version 3.0.1.* Standard. FlexRay Consortium.
[3] Josef Berwanger, Martin Peteratzinger, and Anton Schedl. 2008. FlexRay startet durch - FlexRay-Bordnetz für Fahrdynamik und Fahrerassistenzsysteme (in German). https://www.elektroniknet.de/flexray-startet-durch-1127.html. [Online: accessed 25-May-2019].
[4] Stephen Checkoway, Damon McCoy, Brian Kantor, Danny Anderson, et al. 2011. Comprehensive experimental analyses of automotive attack surfaces.. In *USENIX Security Symposium*, Vol. 4. San Francisco, 447–462.
[5] Don Coppersmith and Markus Jakobsson. 2002. Almost optimal hash sequence traversal. In *International Conference on Financial Cryptography*. Springer, 102–119.
[6] Quynh Dang. 2012. *Recommendation for Applications Using Approved Hash Algorithms.* Technical Report. National Institute of Standards and Technology.
[7] Z. Gu, G. Han, H. Zeng, and Q. Zhao. 2016. Security-Aware Mapping and Scheduling with Hardware Co-Processors for FlexRay-Based Distributed Embedded Systems. *IEEE Transactions on Parallel and Distributed Systems* 27, 10 (Oct 2016), 3044–3057.
[8] Donggang Liu and Peng Ning. 2007. *Security for wireless sensor networks.* Vol. 28. Springer Science & Business Media.
[9] Paul Milbredt, Martin Horauer, and Andreas Steininger. 2008. An investigation of the clique problem in FlexRay. In *2008 International Symposium on Industrial Embedded Systems.* IEEE, 200–207.
[10] Charlie Miller and Chris Valasek. 2015. Remote exploitation of an unaltered passenger vehicle. *Black Hat USA* 2015 (2015), 91.
[11] Pal-Stefan Murvay and Bogdan Groza. 2018. Practical Security Exploits of the FlexRay In-Vehicle Communication Protocol. In *International Conference on Risks and Security of Internet and Systems.* Springer, 172–187.
[12] Dennis Nilsson, Ulf Larson, Francesco Picasso, and Erland Jonsson. 2009. A first simulation of attacks in the automotive network communications protocol flexray. In *Proceedings of the International Workshop on Computational Intelligence in Security for Information Systems CISIS'08.* Springer, 84–91.
[13] A Perrig, D Song, R Canetti, JD Tygar, and B Briscoe. 2005. *Timed Efficient Stream Loss-Tolerant Authentication (TESLA): Multicast Source Authentication Transform Introduction.* RFC 4082. IETF. 1–22 pages. https://tools.ietf.org/html/rfc4082
[14] Von Chait Phagoo, Jim Lawlis, Payam Naghshtabrizi, Emmerich Fuchs, and Gerald Freiberger. 2011. Prototyping von FlexRay-Applikationen bei der Ford Motor Company - FlexRay für Ford (in German). https://www.elektroniknet.de/elektronik-automotive/sonstiges/prototyping-von-flexray-applikationen-bei-der-ford-motor-company-77276.html. [Online: accessed 25-May-2019].
[15] Dominik Püllen, Nikolaos Athanasios Anagnostopoulos, Tolga Arul, and Stefan Katzenbeisser. 2019. Security and Safety Co-Engineering of the FlexRay Bus in Vehicular Networks. In *Proceedings of the International Conference on Omni-Layer Intelligent Systems (COINS '19).* ACM, New York, NY, USA, 31–37.
[16] C. Szilagyi and P. Koopman. 2009. Flexible multicast authentication for time-triggered embedded control network applications. In *2009 IEEE/IFIP International Conference on Dependable Systems Networks.* 165–174.
[17] Paula Vasile, Bogdan Groza, and Stefan Murvay. 2015. Performance Analysis of Broadcast Authentication Protocols on CAN-FD and FlexRay. In *Proceedings of the WESS'15: Workshop on Embedded Systems Security (WESS'15).* ACM, New York, NY, USA, Article 7, 8 pages.
[18] Ronghua Wang, Wenliang Du, Xiaogang Liu, and Peng Ning. 2009. ShortPK: A short-term public key scheme for broadcast authentication in sensor networks. *ACM Transactions on Sensor Networks (TOSN)* 6, 1 (2009), 9.
[19] A. Wasicek, C. El-Salloum, and H. Kopetz. 2011. Authentication in Time-Triggered Systems Using Time-Delayed Release of Keys. In *2011 14th IEEE International Symposium on Object/Component/Service-Oriented Real-Time Distributed Computing.* 31–39.
[20] Marko Wolf, André Weimerskirch, and Christof Paar. 2004. Security in automotive bus systems. In *Workshop on Embedded Security in Cars.*
[21] H. Zeng, M. Di Natale, A. Ghosal, and A. Sangiovanni-Vincentelli. 2011. Schedule Optimization of Time-Triggered Systems Communicating Over the FlexRay Static Segment. *IEEE Transactions on Industrial Informatics* 7, 1 (Feb 2011), 1–17.