Performance analysis of broadcast authentication protocols on CAN-FD and FlexRay

Paula Vasile
paula.vasile10@gmail.com

Bogdan Groza
bogdan.groza@aut.upt.ro

Stefan Murvay
stefan.murvay@gmail.com

Politehnica University of Timisoara, Faculty of Automatics and Computers
Department for Automatics and Applied Informatics
Bd. V. Parvan nr. 2, Timisoara, Romania

ABSTRACT

In the light of the numerous reported attacks, designing cryptographic protocols for in-vehicle embedded networks was a constant preoccupation in the past few years. While several research proposals appeared, a concrete performance analysis of such protocols over a realistic network configuration is still absent from the literature. In this work we address the performance for various authentication protocols that were recently proposed for the two most prominent vehicular buses: FlexRay and CAN-FD. While a real-world vehicular network is still out of reach for our work, we achieve a first step in this direction by using a CANoe based simulation for these protocols over state-of-the-art automotive buses. This allows us to draw a more realistic perspective on the efficiency of existing proposals for bus authentication. Our results suggest that sharing symmetric keys between groups of nodes is the most realistic proposal as it creates a balance between bandwidth efficiency and security level.

Categories and Subject Descriptors

[**Security and privacy**]: Security in hardware—*Embedded systems security*

General Terms

Security

Keywords

broadcast authentication, embedded networks,
CAN-FD, FlexRay

1. INTRODUCTION

Contemporary vehicles are the result of an evolutionary process that augmented mere mechanical devices with complex electronics and intricate software counterparts. Recent vehicles have dozens of ECUs (Electronic Control Units) that implement a plethora of functions dedicated for safety critical tasks, e.g., braking and stability control, or mere informational purposes to make a more pleasurable driving experience. Of course, behind these miniature computers called ECUs, a complex network infrastructure needs to be deployed that connects these ECUs via cables and unavoidably exposes information to the outsiders via various ports, e.g., OBD (On-Board Diagnostics), or even wireless channels, e.g., WiFi, Bluetooth, 3G, etc. There are little doubts that in terms of attacks and defenses vehicular networks will evolve in a similar manner to computer networks. A solid proof for this are the countless attacks that were published in the recent years [9], [2].

WESS'15, October 04-09, 2015, Amsterdam, Netherlands
Copyright 2015 ACM ISBN 978-1-4503-3667-3/15/10 ...$15.00
DOI: http://dx.doi.org/10.1145/2818362.2818369 .

The academic research community was quick to react with various proposals for assuring security on vehicular buses (these are all surveyed in a forthcoming section). However, a comprehensive performance analysis of these solutions on a real-world vehicular network is still missing. The reason behind this is quite simple, in-vehicle infrastructures are still subject to many proprietary solutions and architectural details that are not fully accessible to academic researchers. In this work we take a first step in this direction by using the industry standard CANoe tool in order to simulate a realistic state-of-the-art network based on FlexRay and CAN-FD.

1.1 State-of-the-art in-vehicle buses: FlexRay and CAN-FD

The reasons for choosing these buses are clear: CAN is the most common bus inside cars and FlexRay was designed as its successor. Due to inherent expenses, FlexRay had not yet entered in all vehicles and a direct successor of CAN also emerged, i.e., CAN-FD (Controller Area Network with Flexible Data-rate). We give next a brief description of these two communication layers.

FlexRay is an automotive communication protocol developed by the FlexRay Consortium which gathers important players in the automotive industry. It was designed as a faster and more reliable alternative to other automotive communication protocols existing at that time, e.g., CAN or LIN. The protocol supports data rates up to 10 Mbit/s and a data payload length of 254 bytes. The data is transmitted between ECUs in the form of frames which have the structure presented in Figure 1. The bit-length of the fields are displayed below, the only exception being the data field for which the length is represented in bytes. FlexRay frames can be either time-triggered (static frames) or event-triggered (dynamic frames). FlexRay accomplishes the use of both static and dynamic frames by employing a predefined communication cycle (Figure 1) which defines specific segments

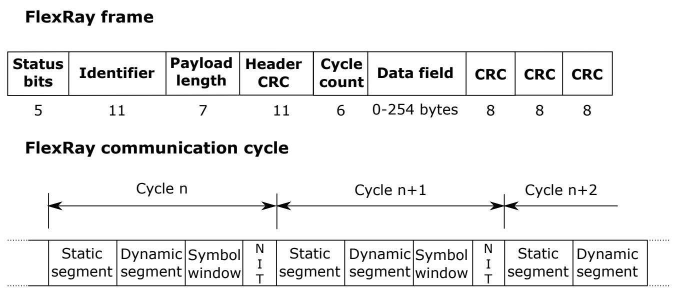

Figure 1: FlexRay frame and communication cycle

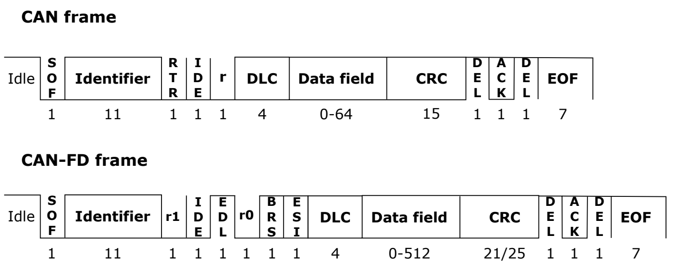

Figure 2: CAN frame versus CAN-FD frame

for static and dynamic data. The communication cycle contains an additional Symbol Window which typically holds network maintenance data and signals for network start-up. A Network Idle Time is used at the end of each communication cycle to maintain synchronization. The space for static and dynamic data is configured during the network design step. In our experimental FlexRay network, all the FlexRay frames were configured as static, so they are being sent cyclically with variable recurrences.

As a downside, FlexRay is more expensive and complex. Therefore, it is mainly used in the safety critical applications of an automobile where strict message transmission timings are enforced such as the power train domain.

CAN-FD (CAN with Flexible Data Rate) [16] is a protocol designed by Robert Bosch GmbH to extend the standard CAN protocol [15]. It was developed as a result of the increasing demand for higher bandwidth. Figure 2 illustrates the differences between a standard CAN frame and a CAN-FD frame. The length in bits of each field is indicated below the frame representation. While the standard CAN only supports payloads of maximum 8 bytes, CAN-FD can accommodate up to 64 bytes of data. For achieving data rates higher than the 1Mbit/s limit in standard CAN, the CAN-FD specification introduces the possibility to switch the baudrate of the data field by setting the BRS (Bit Rate Switch) bit. Speeds of up to 8 MBit/s can be obtained for the data phase, resulting in an average speed for the entire frame of 2.5 MBit/s [7]. For backward compatibility CAN-FD allows the transmission of standard CAN frames. This is controlled by the EDL (Extended Data Length) bit which should be set to a dominant state for CAN. Recently, CAN-FD received the ISO status [8] and was subject to small changes in order to increase its reliability, i.e., the CRC field was extended for improved detection of communication errors [12]. These modifications do not affect the results from this work. In our experimental network, all the CAN-FD messages are configured to be sent cyclically with a recurrence of 5 up to 80 ms.

2. PROPOSED PROTOCOLS, A SYNTHETIC COMPARISON

This section contains an overview of recently proposed protocols for assuring authentication in automotive networks. A synthetic performance comparison of the presented protocols follows, we then take this comparison to the experimental level on our experimental network topology with the help of the CANoe simulation.

2.1 Brief overview of recent proposals

We now enumerate some of the recent proposals, subsequently, they are revisited from a key allocation perspective which is decisive for the performance evaluation that follows.

1. *Voting schemes* [17], [18], [19] require nodes to be present and vote on the authenticity of messages. This does not only require nodes to be present on the bus but also for a message to accumulate a sufficient number of votes which introduces additional delays. Moreover, the receive history of the nodes has to match. These restrictions seem not to be appropriate for in-vehicle networks, a reason for which we do not consider the scheme for our analysis.

2. *TESLA* [14] is an efficient broadcast protocol employed in wireless sensor networks [13]. Implementing this protocol on the CAN bus was considered in [4]. There is a significant drawback in adopting it for in-vehicle networks and this is the authentication delay which is always present in such protocols. The problem is not only that messages can be authenticated with some delay, e.g., usually 1–10 ms, but the node has to buffer all messages and authentication tags received in this time slot for subsequent authentication (this raises memory concerns). However, insofar TESLA is the only way for assuring full broadcast authentication without more expensive public-key cryptographic primitives, e.g., digital signatures. For this reason, we consider this protocol as a candidate for our analysis.

3. *CANAuth* is a scheme that proposes the use of an ID-oriented key allocation [20]. The drawback behind this scheme is that the IDs which are broadcast on a CAN bus are generally too numerous to allow the allocation of a key for each ID. Moreover, sharing the key with nodes that are allowed to receive a particular ID (in order to be able to authenticate the messages) exposes the security key to nodes that can be potentially malicious (e.g., a corrupted diagnosis tool). However, the ID-oriented allocation nicely fits the specification of CAN. Moreover, by adding a single authentication tag to each of the broadcast message this scheme provides a baseline for efficiency as it leads to a simple one-tag-per-message authentication. As proposed in [20] the protocol was intended for CAN+, a variation of CAN that currently does not exist in practice. But the protocol can be ported as is on any other layer, e.g., CAN or CAN-FD.

4. *MaCAN* is proposed in [6]. The protocol has the merit

of being a realistic proposal, it employs shared keys between nodes and CBC-MAC based authentication tags. There are not enough details on how to share the keys between the nodes, except for the fact that these should be shared in a pair-wise manner (this is not suitable for higher number of nodes). The authors of MaCAN [6] suggest that nodes can be grouped under the same key if they share the same trust level which will lead to a reasonable number of keys, but no practical insights are given on how to decide the trust level.

5. *LiBrA-CAN* [5] proposes a more demanding key allocation procedure which mixes keys between groups of nodes (rather then sharing keys pair-wisely). The proposal is more demanding from a computational point of view, but it offers a higher security level in case when adversaries are in minority (a likely scenario for in-vehicle networks).

6. *CaCAN* [10] introduces a centralized view over the authentication process. In this protocol a central node verifies the authentication tag of each frame and if authentication fails, the frame is discarded with error flags. This procedure has the merit of requiring a single monitor node with higher computational power. However, an adversary that removes this node from the bus can take full control of the bus since there is no way for the other nodes to decide if a frame is authentic or not.

7. Other symmetric key schemes were discussed in [22], [21], [1], [11] but the way in which keys and tags are allocated seems to fit in one of the previous paradigms and thus we do not consider them as separate cases for our simulation.

2.2 Revisiting protocols from a keying perspective

While in the previous subsection we encountered several proposals, the crux of the problem (and the difference between the previous schemes) comes from the way in which the keys are distributed between the nodes. It is clear that only *symmetric key* primitives can be used, due to restrictions on computational power and bandwidth, and thus the number of authentication tags (which determines the bus-load) comes from the way in which keys are allocated. We now revisit the previous proposals and classify them based on the keying procedure they employ. Mutatis mutandis, all of the previous schemes fit in one of the following four paradigms:

1. *Single authentication key* is the simplest keying procedure in which all frames are authenticated with a single key. In this case, all senders and receivers must be in possession of the authentication key and if one of them is corrupted all security is lost. Since each frame carries a single tag, this protocol provides a baseline for the bus-load (from this perspective both CANAuth [20] and CaCAN [10] fit into this paradigm). *ID-oriented keying* is a variation of this in which each frame carries a single authentication tag, but the key to compute this tag is unique for each of the IDs. This procedure is explored in CANAuth [20]. As already noted in previous work, usually there are too many IDs to have a unique key for each of them, but this requirement can be relaxed by having a unique key for each group of frames that is selected based on a predefined mask. From the bus-load perspective this protocol has identical requirements to the previous (a single tag again) and thus it matches the baseline for the bus-load when a single authentication key is used.

2. *Pairwise keying* is the rather natural way in which unique authentication keys are shared by each distinct pair of nodes. This procedure is also employed in MaCAN [6]. However, in case of n nodes this leads to $\frac{n(n-1)}{2}$ keys and $n-1$ authentication tags. For higher number of nodes, the overhead is unlikely to be handled by the available bandwidth and computation power available in automotive networks. For this reason, in related work, e.g., [6], it was already suggested that nodes that share the same trust level can use the same secret key for authentication. In some sense, this opens door for the next procedure.

3. *Group keying* is an improvement over pairwise key sharing that allocates keys over groups of nodes. The main ideea is to build groups that have an intermediate size between 2 (which corresponds to pairwise keying) and n (which corresponds to having a single authentication key in case of n nodes). The security level is higher if malicious nodes are in minority, the procedure is explored in LiBrA-CAN [5]. Worst than in the previous case, the number of keys and tags grows exponentially, but again if more nodes are grouped under the same key (as in the case of pair-wise keying) then the number of keys stays low.

4. *TESLA-like keying* requires authentication keys to be broadcast in a periodic manner, i.e., at fixed time intervals. This means that besides the regular frames that are sent over the network a frame containing the key is released at fixed intervals. However, the problem of the protocol is not necessarily in the additional overhead but in buffering the received frames until the authentication key is received and in the intrinsic authentication delay.

For the case of shared keys, i.e., all protocols except TESLA, in order to draw a synthetic comparison we need to fix the following parameters for a setup with n receivers in groups of size g:

- the total number of keys:

$$\mathcal{K} = \binom{n}{g},$$

- the total number of keys stored on a single node which is also the number of tags computed by the node when sending a message to all of the other nodes:

$$\mathcal{K}_{\text{send}} = \binom{n-1}{g-1},$$

- the total number of tags intended for a single receiver which is also the computational load on the receiver side:

$$\mathcal{K}_{\mathrm{recv}} = \binom{n-2}{g-2},$$

- the fraction of tags intended for a single node out of the total number of tags:

$$\mathrm{F}_{\mathrm{recv}} = \binom{n-2}{g-2}\binom{n-1}{g-1}^{-1} = \frac{\mathcal{K}^{\mathrm{node}}_{\mathrm{recv}}}{\mathcal{K}^{\mathrm{node}}_{\mathrm{send}}},$$

- the size of the tag for a security level of ℓ bits:

$$\mathrm{S} = \ell \binom{n-2}{g-2}^{-1}\binom{n-1}{k-1} = \ell \cdot \mathrm{F}^{-1}_{\mathrm{recv}},$$

- the fraction of uncorrupted tags for a single receiver in case of m corrupted nodes:

$$\mathrm{F}^{\mathrm{corr}}_{\mathrm{recv}} = \frac{\binom{n-2-m}{g-2}}{\binom{n-1}{g-1}},$$

- the security level in case of m corrupted nodes which is the fraction of uncorrupted security bits for a single receiver:

$$\ell^{\mathrm{corr}} = \mathrm{S} \cdot \mathrm{F}^{\mathrm{corr}}_{\mathrm{recv}}.$$

In this formalism, the case when the size of the subgroup is $g = n$ corresponds to the case of a single authentication key while $g = 2$ corresponds to the case of pair-wise key sharing. In terms of bandwidth, the ID-oriented keying and TESLA-like keying overlaps with the case of a single authentication key. We can now draw a synthetic comparison.

In Figures 3 and 4 we depict the size of the tag as well as the remaining uncompromised bits in case of 1 corrupted node with $n = 8$. It is easy to note that when a single key is used, i.e., $n = 8, g = 8$, the size of the resulting authentication tag is kept at a minimum, i.e., 128 bits for a security level of 128 bits. In the same case however, if a single node is corrupted the number of uncorrupted bits drops to 0; since all nodes share the same key. Sharing the keys pair-wisely leads to no security loss in case of 1 corrupted node, as each two nodes share a distinct key, however for a security level of 128 bits the tag quickly grows to 896 bits (the plot is truncated at 250 bits since tags larger than this are clearly not suited for real-time communication on an embedded network). Finally, having groups of size 3 or 4 gives a nice trade-off between these values.

Figure 5 compares the number of keys on each node, tags computed by each node and tags verified by each node. For the extreme case of group size 2 and 8 these values are the lowest but with the aforementioned disadvantage that either the tag is too large, i.e., $g = 2$, or security is lost when one node is compromised, i.e., $g = 8$.

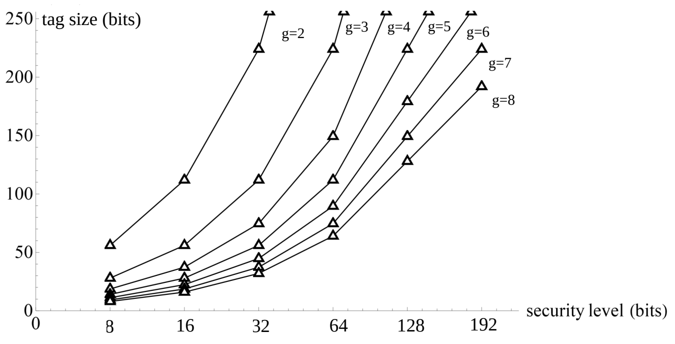

Figure 3: Tag size for a security level of 8 up to 192 bits with n=8 and g=2,3...8

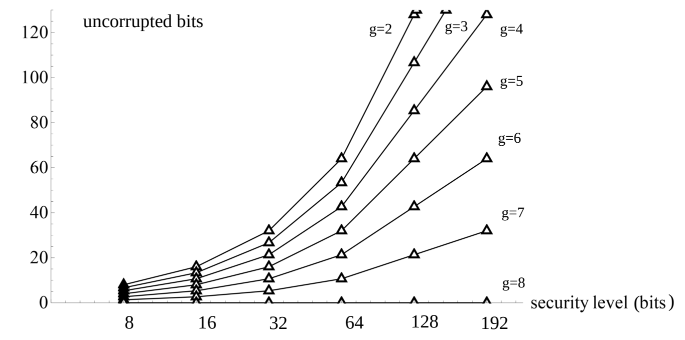

Figure 4: Uncorrupted bits in case of 1 corrupted node for a security level of 8 up to 192 bits with n=8 and g=2,3...8

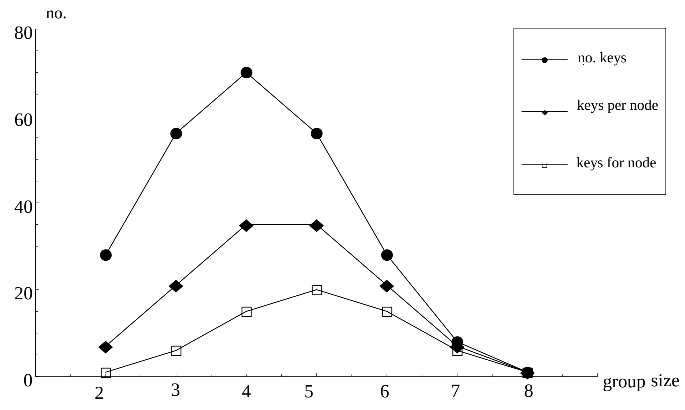

Figure 5: Comparison on the number of keys, number of computed tags and number of verified tags with n=8 and g=2,3...8

3. EXPERIMENTAL EVALUATION

We begin by discussing the topology of the network and some issues regarding the protocol, then we proceed to the experimental results.

3.1 Network topology

The network topology discussed below stays at the core of our CANoe based simulation. The networks were designed to comply with real-world in-vehicle networks used by the automotive industry, in order to obtain experimental results

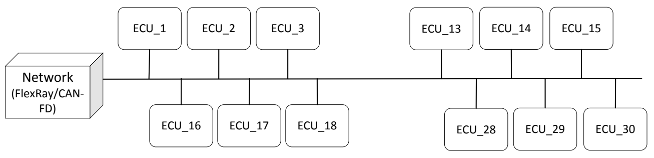

Figure 6: Network topology

that are as close as possible to the real-world behaviour.

Figure 6 depicts the experimental network topology consisting of 30 ECUs. All nodes are interconnected using the same bus type, thus, separate simulations were built for each bus type (FlexRay and CAN-FD) on the same topology. The number of ECUs was chosen to represent the maximum number of ECUs specified by the High Speed ISO 11898 Standard for a CAN network having a maximum signalling rate of 1Mbps and a bus length of 40 m [3]. By relying on the same topology for both FlexRay and CAN-FD, we want to give a crisp image on the difference in performance for each bus type under the same authentication protocol. This of course complements the image over the protocols performance that can be individually drawn for each bus type.

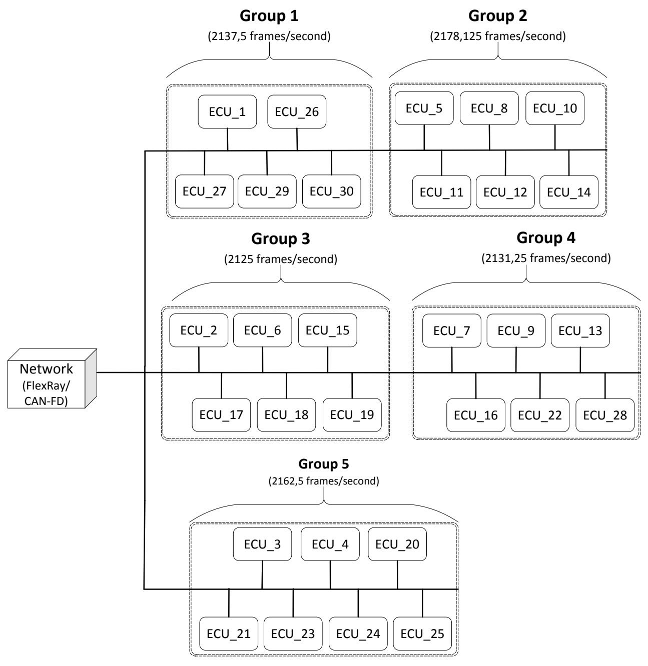

Figure 7: Grouping of ECUs into clusters

Protocols relying on *group keying* require special attention as nodes need to be further grouped. The 30 ECUs forming the network presented in Figure 6 are organized in 5 *clusters*. Each group should gather nodes that share the same trust level as they will be in possession of the same key. Since there are not many hints on how this decision should be taken (in the absence of manufacturer specifications), the allocation scheme that we opted for was to obtain a balanced number of frames transmited by each group (this can be seen at least as a secondary, more practical criteria). The busload based grouping is depicted in Table 1. Thus, each of the 5 clusters generates approximately the same load on the bus. Figure 7 shows the way in which we now group the ECUs on our network.

Group	ECU	Frames/ second/ ECU	Frames/ second/ Group
1	ECU_29	600	
	ECU_27	500	
	ECU_30	425	
	ECU_26	200	
	ECU_1	412.5	2137.5
2	ECU_10	625	
	ECU_11	325	
	ECU_12	425	
	ECU_5	200	
	ECU_8	403.125	
	ECU_14	200	2178.125
3	ECU_15	406.25	
	ECU_17	225	
	ECU_18	362.5	
	ECU_6	425	
	ECU_19	206.25	
	ECU_2	500	2125
4	ECU_7	206.25	
	ECU_16	350	
	ECU_22	106.25	
	ECU_13	350	
	ECU_9	400	
	ECU_28	750	2162.5
5	ECU_20	206.25	
	ECU_21	325	
	ECU_4	200	
	ECU_23	425	
	ECU_24	362.5	
	ECU_25	212.5	
	ECU_3	400	2131.25

Table 1: Busloads for each group of ECUs

3.2 Comparative results

The comparative evaluation of the protocols can account for two distinct characteristics: the bandwidth (which depends on the number and size of the authentication tags) and the computational load on each ECU (which depends on the number of tags that each sender/receiver has to compute/verify each second). Our bus simulation is of course focused on the first aspect. The computational load is not to be neglected, however since hardware implementations are available for cryptographic primitives, e.g., AES, and message authentication primitives, e.g., CBC-MAC, can be straight-forwardly derived, the concerns on computational power are little. As a rough baseline, a hardware implementation will allow for an authentication tag to be computed in several micro-seconds, while even a software implementation on a high-grade ECU, e.g., Infineon TriCore, will cost in the order of a dozen micro-seconds, allowing each node to process tens of thousands of authentication tags each second. This quantity however, cannot be handled by the available bandwidth if each node is assumed to send the maximum number of tags it can compute. If needed, computational concerns can be subject of future work for us, for the moment the maximum computational load seems to stay below 10.000 tags/second as will be further shown.

In what follows, we briefly discuss characteristics of each protocol in the context set by our experiments.

1. *Single authentication key* implies a single key and tag to be added to each frame. The simulation was performed by adding three different tag sizes: 32, 64 and 128 bit tags. This results in at most 128 bits/frame increase in the payload which is in fact the minimum load for performing authentication. The *computational load* of a receiving ECU increases proportionally with the number of sent/received frames per second.

2. *Pairwise keying* requires the number of authentication tags to be equal to the number of ECUs that are receiving the frame. The worst case scenario is encountered if an ECU broadcasts a frame to all the other ECUs on the network. In our experimental network with 30 ECUs, this translates to 29 authentication tags/frame. Table 2 presents the payload increase for 32, 64, respectively 128 bit tags, for networks with 30, 20 and 10 ECUs. We can immediately rule out any simulation of pairwise keying on our experimental network since the necessary overhead, even for 32-bit tags, exceeds the maximum payload of a CAN-FD (64 bytes) and, sometimes, of a FlexRay frame (256 bytes). Therefore, measuring the busload is not of interest in this case since this option is not feasible for practice.

Tag size	**No. of ECUs in network**	**Overhead [bits]**	**Overhead [bytes]**
32	30	928	116
64	30	1856	232
128	30	3712	464
32	20	608	76
64	20	1216	152
128	20	2432	304
32	10	288	36
64	10	576	72
128	10	1152	144

Table 2: Frame overhead in case of pairwise keying

3. *TESLA-like keying* requires for each sender (ECU on the network) a new frame containing a 32, 64 or 128 bit authentication tag that is periodically broadcast. To reduce the overhead, a master oriented scenario can be imagined where a unique master-node sends the authentication frame, but this approach is not compatible with the CAN standard [4]. We ran simulations for all tag sizes and key release intervals of 10, 20, 40 and 80 ms. Although there is no significant increase in the payload of the existing frames, the addition of new frames leads to a more considerable increase in the busload (depending on the release interval). In terms of computational and memory load, it gets worst, as each ECU needs to store the frames until the authentication keys are released. As expected, this number increases proportionally with the key release interval as shown in Figure 8 (the same values also dictate the computational load measured in computed tags / second).

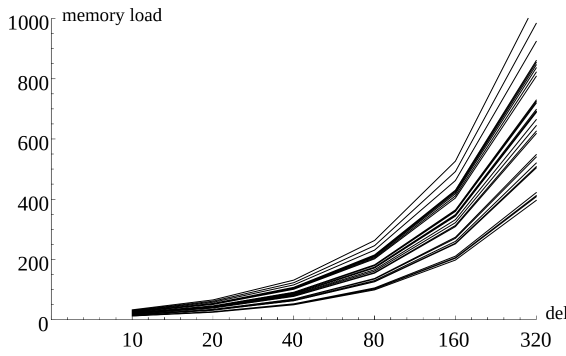

Figure 8: Dependence of memory load (in terms of frames waiting for authentication) with key release interval on each of the 30 ECUs with TESLA

4. *Group keying* is a general concept which provides a more realistic deployment method by reducing the number of keys. For our 30-ECU experimental network, by grouping the ECUs into 5 clusters the payload increase is considerably smaller than for pairwise keying. The 5 clusters from Figure 7 can be viewed as 5 ECUs with that can be grouped in pairs of 2, 3 or 4. Therefore, the maximum number of authentication tags added to a frame will be the number of shared keys between the sending cluster and the receiving clusters. This is summarized in Table 3. With a maximum number of 6 authentication tags added to a frame, group keying can be successfully simulated on our experimental network using authentication tags of 32 and 64 bits. The additional payload seems not to be significant for the network's overall busload. The multiple authentication tags lead to an increase in the computational load, one ECU having to compute even 6 tags for the same frame, but this can be alleviated by hardware implementation. Table 3 indicates that gathering clusters with 4 nodes that share the same key brings the optimal compromise between provided security, payload and computational load out of all the considered authentication methods.

Tables 4, 5 and 6 summarize the performance data for the 4 authentication paradigms in terms of authentication payload (Table 4), computational load (Table 5) and recorded busload (Table 6), the latter being measured by the CANoe simulation tool, separately for the FlexRay and CAN-FD experimental networks.

The payload of each frame increases from several bytes (4–16) per frame in case of TESLA or single keys up to several dozens bytes in case of group keying (24–96). This increase can be handled by the network at a busload between 43.36% and 71.61% as shown in Table 6. The computational load varies from a little more than 1.000 tags/second up to almost 10.000 tags/second but these loads should be easily handled by hardware implementation on modern ECUs.

Finally, from a busload perspective, TESLA and group keying seem to stay close in terms of performance. The more efficient single authentication key has only a 10%–30%

No. of clusters sharing a key	No. of receiver clusters	No. of added tags
2	1	1
	2	2
	3	3
	4	4
	5	4
3	1	3
	2	5
	3	6
	4	6
	5	6
4	1	3
	2	4
	3	4
	4	4
	5	4

Table 3: Number of tags added to a frame when applying *Group keying*

advantage in the busload which is unclear if it can prove worthy given the fragile security (compromising a single key lead to complete security loss). We can observe that, for the FlexRay network, the busload changes only when additional frames are introduced, the addition of data bytes to the existing frames has little effect. This is due to the fact that the FlexRay network has a global payload size - in our case 56 bytes - which is set when configuring the network. Unlike the CAN-FD, which allows the adjustment of each frame's size, the FlexRay frames are all sent with the configured global payload size, even though part of the payload may not contain any data. The increase in busload is also suggested in Table 7.

Authentication protocol	Tag size [bits]	Max. payload [bits]	Max. payload [bytes]
Single authentication key	32	32	4
	64	64	8
	128	128	16
Pairwise keying	32	928	116
	64	1856	232
	128	3712	464
TESLA-like keying	32	32	4
	64	64	8
	128	128	16
Group keying	32	192	24
	64	384	48
	128	768	96

Table 4: Payload for the considered authentication protocols on all tag sizes

4. CONCLUSION

The conclusions can be directly drawn from the experimental section. Pairwise keying leads to a minimum computational load, an ECU having to compute only one authentication tag for a received frame, but table 2 proves that it is not a feasible authentication solution for networks with more than 10 ECUs, especially in case of a CAN-FD network having maximum 64 bytes of payload due to the increase payload to each frame. TESLA-like keying, although bringing a minimum increase in the payload and computational load, is not a realistic solution for real-time automotive systems where all the data contained in the frames has to be processed immediately when received and cannot wait for the release of the authentication key (buffering also represents a significant problem). Single authentication key may seem a viable solution, with little payload and busload, but its main drawback is the low level of security it provides. Compromising only one node of the network will compromise the entire network traffic. Group keying, although having the highest computational load from all the presented protocols, also ensures a higher security level when compromised nodes are in minority. Since every frame is authenticated with multiple tags, compromising only one node of the network does not compromise the whole network and the malicious node can be immediately identified.

While choosing a specific protocol depends on standardization that is done by the industry, our results suggest that *group keying* should be the method of choice for in-vehicle networks.

Authentication protocol	MIN comp. load [tags/s]	MAX comp. load [tags/s]	AVG comp. load [tags/s]
Single authentication key	1237.5	3290.625	2127.8125
TESLA-like keying	1237.5	3290.625	2127.8125
Group keying	3712.5	9871.875	6393.4375

Table 5: Computational load for the considered authentication protocols

Authentication protocol	Recorded busload: CAN-FD [%]	Recorded busload: FlexRay [%]
Baseline	43.36	58.55
Single authentication key	48.97	58.57
TESLA-like keying 10 ms	64.97	71.61
TESLA-like keying 20 ms	57.40	65.89
TESLA-like keying 40 ms	53.62	61.56
TESLA-like keying 80 ms	51.73	59.41
Group keying (groups of 2)	68	58.57
Group keying (groups of 3)	82.21	58.57
Group keying (groups of 4)	70.51	58.57

Table 6: Recorded busload on CAN-FD and FlexRay

Authentication protocol	Increase of busload: CAN-FD [%]	Increase of busload: FlexRay [%]
Single authentication key	13	0.03
TESLA-like keying 10 ms	50	22.3
TESLA-like keying 20 ms	32.38	12.46
TESLA-like keying 40 ms	23.66	5.14
TESLA-like keying 80 ms	19.3	1.46
Group keying (groups of 2)	56.82	0.03
Group keying (groups of 3)	89.59	0.03
Group keying (groups of 4)	62.61	0.03

Table 7: Increase in busload on CAN-FD and FlexRay

5. ACKNOWLEDGMENTS

We are grateful to Vector Informatik GmbH in Vienna (Austria) and to Mr. Mihai Suta for providing us the support for the CANoe tool. We would also like to thank the anonymous referees for helpful comments on our work.

6. REFERENCES

[1] S. Bittl. Attack potential and efficient security enhancement of automotive bus networks using short MACs with rapid key change. In *Communication Technologies for Vehicles*, pages 113–125. Springer, 2014.

[2] S. Checkoway, D. McCoy, B. Kantor, D. Anderson, H. Shacham, S. Savage, K. Koscher, A. Czeskis, F. Roesner, and T. Kohno. Comprehensive experimental analyses of automotive attack surfaces. In *USENIX Security 2011*, 2011.

[3] S. Corrigan. Controller area network physical layer requirements. *Application Report, Texas Instruments*, 2008.

[4] B. Groza and P.-S. Murvay. Efficient Protocols For Secure Broadcast In Controller Area Networks. *Industrial Informatics, IEEE Transactions on*, 2012.

[5] B. Groza, S. Murvay, A. Van Herrewege, and I. Verbauwhede. LiBrA-CAN: a lightweight broadcast authentication protocol for controller area networks. In *Cryptology and Network Security*, pages 185–200. Springer, 2012.

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

[7] F. Hartwich. CAN with flexible data-rate. In *13th International CAN Conference (iCC2012), Hambach, Germany*, 2012.

[8] International Organization for Standardization. *ISO/DIS 11898-1: Road vehicles - Controller Area Network - Part 1: Data link layer and physical signalling*, 2015.

[9] K. Koscher, A. Czeskis, F. Roesner, S. Patel, T. Kohno, S. Checkoway, D. McCoy, B. Kantor, D. Anderson, H. Shacham, and S. Savage. Experimental security analysis of a modern automobile. In *Security and Privacy (SP), 2010 IEEE Symposium on*, pages 447 –462, May 2010.

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

[11] C.-W. Lin, Q. Zhu, and A. Sangiovanni-Vincentelli. Security-aware modeling and efficient mapping for CAN-based real-time distributed automotive systems. *Embedded Systems Letters*, 2014.

[12] A. Mutter. CAN-FD and the CRC issue. *CAN Newsletter 1/2015. CAN in Automation (CiA)*, 2015.

[13] A. Perrig, R. Canetti, D. Song, and J. D. Tygar. SPINS: Security protocols for sensor networks. In *Seventh Annual ACM International Conference on Mobile Computing and Networks (MobiCom 2001)*, pages 189–199, 2001.

[14] A. Perrig, R. Canetti, J. Tygar, and D. X. Song. Efficient authentication and signing of multicast streams over lossy channels. In *IEEE Symposium on Security and Privacy*, pages 56–73, 2000.

[15] Robert BOSCH GmbH. *CAN Specification Version 2.0.*, 1991.

[16] Robert BOSCH GmbH. *CAN with Flexible Data-Rate Version 1.0*, 2012.

[17] C. Szilagyi and P. Koopman. Flexible multicast authentication for time-triggered embedded control network applications. In *Dependable Systems & Networks, 2009. DSN'09. IEEE/IFIP International Conference on*, pages 165–174. IEEE, 2009.

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

[19] C. J. Szilagyi. *Low cost multicast network authentication for embedded control systems*. PhD thesis, Carnegie Mellon University, 2012.

[20] A. Van Herrewege, D. Singelee, and I. Verbauwhede. CANAuth-a simple, backward compatible broadcast authentication protocol for CAN bus. In *ECRYPT Workshop on Lightweight Cryptography 2011*, 2011.

[21] Q. Wang and S. Sawhney. VeCure: A practical security framework to protect the CAN bus of vehicles. In *Internet of Things (IOT), 2014 International Conference on the*, pages 13–18. IEEE, 2014.

[22] S. Woo, H. J. Jo, and D. H. Lee. A practical wireless attack on the connected car and security protocol for in-vehicle CAN. *Intelligent Transportation Systems, IEEE Transactions on*, 2014.