LiBrA-CAN: A Lightweight Broadcast Authentication Protocol for Controller Area Networks

Bogdan Groza[1], Stefan Murvay[1], Anthony van Herrewege[2], and Ingrid Verbauwhede[2]

[1] Faculty of Automatics and Computers, Politehnica University of Timisoara
{bogdan.groza,pal-stefan.murvay}@aut.upt.ro
[2] ESAT/COSIC - IBBT, KU Leuven, Belgium
{anthony.vanherrewege,ingrid.verbauwhede}@esat.kuleuven.be

Abstract. Security in vehicular networks established itself as a highly active research area in the last few years. However, there are only a few results so far on assuring security for communication buses inside vehicles. Here we advocate the use of a protocol based entirely on simple symmetric primitives that takes advantage of two interesting procedures which we call key splitting and MAC mixing. Rather than achieving authentication independently for each node, we split authentication keys between groups of multiple nodes. This leads to a more efficient progressive authentication that is effective especially in the case when compromised nodes form only a minority and we believe such an assumption to be realistic in automotive networks. To gain more security we also account an interesting construction in which message authentication codes are amalgamated using systems of linear equations. We study several protocol variants which are extremely flexible allowing different trade-offs on bus load, computational cost and security level. Experimental results are presented on state-of-the-art Infineon TriCore controllers which are contrasted with low end controllers with Freescale S12X cores, all these devices are wide spread in the automotive industry. Finally, we discuss a completely backward compatible solution based on CAN+, a recent improvement of CAN.

1 Motivation and Related Work

Vehicular network security established itself as an intense research topic in the last few years. Remarkable research papers from Koscher et al. [7] and later Checkoway et al. [4] showed vehicles to be easy targets for malicious adversaries.

While most of previous research was focused on vehicle to vehicle and vehicle to infrastructure communication there seem to be only a few results for assuring security on communication buses inside vehicles. There are several reasons behind this. First, the relevance of security inside vehicles was decisively shown only in the last two years [7], [4]. Second, the design principles used by manufacturers are somewhat out of reach for the academic community, being hard in this way to understand many assertions behind protocol design. Third, which is relevant for our research here, intra-vehicle communication is subject to constraints and specifications that are quite different from other well studied protocols. Most of the approaches advocate the use of secure gateways between different ECUs (Electronic Control Unit) or subnetworks [1], [13] and rely on basic building blocks from cryptography (encryptions, signatures, etc.). However, none of these approaches is meant specifically for assuring broadcast authentication on CAN which is still the most common communication bus in automotives.

In this respect two main results in assuring CAN security can be found so far, one of them is based on the well known TESLA protocol [6] and the other proposes a new paradigm which closely follows CAN specifications [12]. Van Herrewege et al. [12] design their protocol from scratch and clearly note that the constraints of CAN ”eliminate all the authentication protocols published so far”. We do agree with this conclusion in the sense that we believe that standard authentication approaches, may cover only some of the application areas for CAN and new approaches (even non-standard) are needed.

Previous proposals. TESLA like protocols proved to be highly effective in sensor networks [10], [9] and so far are the most efficient alternative for assuring broadcast authentication with efficient Message Authentication Codes (MAC). However, when it comes to CAN bus, this protocol family has one drawback that is critical for automotives: delays, which by the nature of TESLA are unavoidable. The main purpose of the work in [6] is to determine a lower bound on these delays. Delays in the order of milliseconds or below, as shown to be achievable in [6], are satisfactory for many scenarios, but such delays do not appear to be small enough for intra-vehicle communication. There is no obvious way to improve on these delays further. Of course one alternative is in using a bus with a higher throughput, more computational power and better electronic components (e.g., oscillators) but this will greatly increase the cost of components, nullifying in this way the cost effectiveness of CAN. CANAuth [12] is a protocol that has the merit to follow in great detail the specifications of CAN, its security is specifically designed to meet the requirements of the CAN bus. In particular, CANAuth is not intended to achieve source authentication as the authentication is binded to the message IDs and messages may originate from different sources which will be impossible to trace. This fits the specification of CAN which has a message oriented communication. However, a first issue is that the number of CAN IDs is quite high, in the order of hundreds (11 bits) or even millions in the case of extended frames (29 bits) and storing a key for each possible ID does not seem to be so practical. For this purpose, in [12] a clever solution is imagined: the keys are linked with acceptance codes and masks, which fortunately are not numerous. But still, this leads to some security concerns as we discuss next. Traditionally, keys are associated to entities to ensure that they are not impersonated by adversaries, but the effect of associating keys to messages is less obvious. For example, any external tool (assume On-Board Diagnostics (OBD) tools which are wide spread) that is produced by external third parties will have to embed the keys associated for each ID that it sends over or even just listens on CAN. It is thus unclear which keys can be shared with different manufacturers and how or what are the security outcomes for this. Obviously, if a third party device, even an innocuous one designed just as passive receiver, is easier to compromise then all the IDs which it was allowed to send or just receive are equally compromised.

Our proposal. We take advantage of a progressive authentication mechanism, by which only a few bits of the MAC are revealed in each packet to each verifier, and each part of the MAC can be verified by more than one receiver. To achieve this flexible authentication mechanism we base our proposal on two paradigms: key splitting and MAC mixing, the later being an optional procedure to increase security by allowing any node to detect a potential forgery.

Key splitting allows a higher entropy for each mixed MAC that is sent at the cost of loosing some security for groups that contain malicious nodes. In scenarios with high number of nodes, an adversarial majority will be required to break the protocol, while if there are fewer adversarial nodes, the security level is drastically increased. Consequently, this appears to give a flexible and efficient trade-off. This procedure is not new, similar techniques were proposed in the past in the context of broadcast encryption. We could trace this back up to the work of Fiat and Naor [5] but there is a high amount of papers on this subject. However, the constraints of our application in CAN networks are entirely different from related work where this procedure was suggested or used in scenarios such as sensor networks [2], pay-tv [8], etc. The main idea behind such schemes is that groups of k corrupted receivers cannot learn the secret (in settings with $n > k$ users).

In addition to this we exhibit a distinct contribution in the construction of Linearly Mixed MACs which allow us to amalgamate more authentication codes in one via a system of linear equations. This construction has the advantage that if one of the MACs is wrong then this will affect all other MACs and thus the mixed MAC will fail to verify on any of the multiple keys. This increases the chance of a forgery being detected and ultimately it increases the reliability in front of benign nodes that are in possession of a wrong key. To best of our knowledge this procedure is new. The closest work that we could find are the multi-verifier signatures proposed by Roeder et al. [11]. In their work, linear systems of equations are used as well upon message authentication codes but the security properties and goals of their construction are different.

These procedures allow us to design a protocol that is more flexible and efficient. For our setting we assume a reduced number of participants. While indeed ECUs inside cars come from different manufacturers which may or may not be trustworthy, we believe that suspicious ECUs should be limited in number, since the potential insertion of a trapdoor in some component will discredit the public image of the manufacturer too much and it appears to be little or no benefit for this. More, ECUs coming from the same manufacturer should be trustworthy with each other and can use the same shared key (randomly generated at runtime for each (sub)network that they are part of). In this way the number of actual keys needed to assure broadcast security should be more limited than it appears to be on a first sight. In our design we try to take advantage of this assumption, and our approach is more efficient in the case when compromised nodes form only a minority.

2 The Protocol

We begin with a brief overview of the CAN protocol followed by a description of the frame structure employed in our protocol. Then we outline the main authentication scheme which builds upon keys shared between groups of receivers, a procedure which we call key splitting. Further, we discuss some variations of the main scheme that can be used for different trade-offs. Subsequently we introduce a construction which we call Linearly Mixed MAC (LM-MAC) which gives additional security benefits.

2.1 Overview of the CAN Protocol

Controller Area Network (CAN) is a broadcast serial bus. The typical topology consists of a differential bus which connects multiple nodes by two wires (called CAN-H and CAN-L). This is also suggested in Figure 3 which is related to the main version of our protocol. To avoid collisions an arbitration based on message identifiers (29 bits in extended frames and 11 bits in standard frames) is used. Each CAN frame begins with a start bit and is followed by the arbitration field, a control field (6 bits), data bits (0-64), CRC sequence (15 bits), a 2 bit acknowledgment and 7 bits that mark the end of the frame. Additional stuffing bits (distinct in value to the previous bit) are added after each 6 consecutive bits of identical value. This structure is suggested in Figure 1.

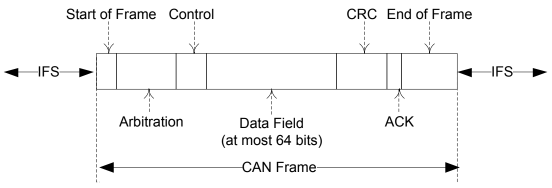

Fig. 1. Structure of a CAN frame

2.2 Frame Structure

As a general procedure, we separate between frames that carry messages and frames that carry authentication tags. This seems to be a correct option due to a widely employed CAN mechanism which is ID filtering that is used to restrict certain frames to arrive to a particular node. While we do want to keep this feature, we want the node to be able to carry additional authentication tasks, e.g., in the case of the two-stage authentication discussed further, a reason for which we intend for the authentication frames to be able to reach the node and thus they may need to have a different ID than the message frame. The last bit of the identifier field specifies whether a frame carries an authentication tag or message. This procedure is employed in our experimental setup while in section 4 we discuss a backward compatible solution which can embed all the authentication information inside the message.

Larger data blocks or authentication tags can be split across multiple frames with the same ID field and counter. Other adjustments can be done at the implementation level. For example, since the ID field is quite short, both the node and window identifiers (which denote the source and the number of the authentication frame) can be moved in the data field. We preferred to place these identifiers in the ID field since it is a frequent choice of developers to place a unique ID for each node in the CAN ID field. But indeed, such an option can affect real-time requirements and for this purpose placing these IDs in the data field is safer. The size of the counter c could be roughly around 20–40 bits but this greatly depends on the bus speed (which determines the number of packets released each second).

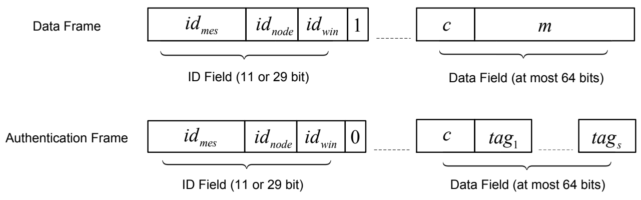

Fig. 2. Data frames and authentication frames

2.3 The Main Scheme: Centralized Authentication

A master oriented communication makes sense since it is practical to have one node with higher computational power that can take care of the most intensive part of the authentication. Figure 3 shows the master node and the slave nodes connected to the bus, it also outlines the keys that are shared between nodes. For the key sharing procedure, all slaves register to the master which distributes the keys.In practice associating nodes to a group and sharing the keys is done by standard techniques, e.g., key-exchange protocols, we do not insist on this since such issues are straight-forward to solve.

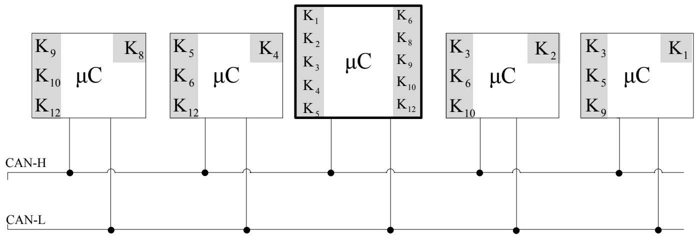
K_9 K_{10} K_{12} K_8 µC
K_5 K_6 K_{12} K_4 µC
K_1 K_2 K_3 K_4 K_5 µC K_6 K_8 K_9 K_{10} K_{12}
K_3 K_6 K_{10} K_2 µC
K_3 K_5 K_9 K_1 µC
CAN-H
CAN-L

Fig. 3. Master and slave microcontrollers (μC) in a setting for centralized authentication

In the main scheme we make use of Mixed Message Authentication Codes (M-MAC) which amalgamate more MACs into one. Here we give an abstract definition for this construction while in a forthcoming section we provide a more elaborate instance with additional security properties. Indeed, the easiest way to build an M-MAC is simply by concatenating multiple tags, such a construction is fine for our protocol and can be safely embodied in the main scheme (still, we can achieve more security with the LM-MAC introduced in an upcoming section).

Construction 1. *(Mixed Message Authentication Code) A mixed message authentication code* M-MAC *is a tuple* (Gen, Tag, Ver) *of probabilistic polynomial-time algorithms such that:*

1. $\mathbb{K} \leftarrow \mathsf{Gen}(1^\ell, s)$ *is the key generation algorithm which takes as input the security parameter* ℓ *and set size* s *then outputs a key set* $\mathbb{K} = \{k_0, k_1, ..., k_s\}$ *of* s *keys,*
2. $\tau \leftarrow \mathsf{Tag}(\mathbb{K}, \mathbb{M})$ *is the* MAC *generation algorithm which takes as input the key set* $\mathbb{K}$ *and message tuple* $\mathbb{M} = (m_0, m_1, ..., m_s)$ *where each* $m_i \in \{0,1\}^*$ *then outputs a tag* τ *(whenever needed, to avoid ambiguities on the message and key, we use the notation* $\mathsf{M\text{-}MAC}_{\mathbb{K}}(\mathbb{M})$ *to depict this tag),*
3. $v \leftarrow \mathsf{Ver}(k, m, \tau)$ *is the verification algorithm which takes as input a key* $k \in \mathbb{K}$*, a message* $m \in \{0,1\}^*$ *and a tag* τ *and outputs a bit* v *which is* 1 *if and only if the tag is valid with respect to the key* k*, otherwise the bit* v *is* 0*. For correctness we require that if* $k \in \mathbb{K}$ *and* $m \in \mathbb{M}$ *then* $1 \leftarrow \mathsf{Ver}(k, m, \mathsf{Tag}(\mathbb{K}, \mathbb{M}))$*.*

The centralized scheme is summarized by the next construction. For simplicity of the exposition, since the main scheme is used to authenticate the same message to all nodes (rather than authenticate a tuple of messages as in the cummulative authentication scheme), we replace $\mathbb{M}$ with a simple array that points out the values that are authenticated, e.g., $id_{node}, id_{win}, c, m$, etc. Obviously in this case the M-MAC receives as input a message tuple of s identical messages.

Construction 2. *(Centralized Authentication) Given a mixed message authentication code algorithm* M-MAC *for some security parameter* ℓ*, size* s *and a group of* n *nodes, we define protocol* $\mathsf{CN\text{-}CAN\text{-}LiBrA}_{\mathcal{M},\mathcal{S}^*}(\mathsf{M\text{-}MAC}, \ell, s, n, b, w)$ *as the following set of actions for the master* $\mathcal{M}$*:*

1. $\mathsf{Setup}(\ell, n, s)$ *on which master* $\mathcal{M}$ *generates all subsets of* s *slaves out of* n *slaves, let* $t = \binom{n}{s}$ *be the number of subsets, and randomly picks* t *keys, each of* ℓ *bits, then places them in the keyset* $\mathbb{K}_{\mathcal{M}} = \{k_1, k_2, ..., k_t\}$*. Subsequently master* $\mathcal{M}$ *uses a secure channel to send each node the corresponding keys (alternatively these keys can be distributed in an off-line manner). Let* $\mathbb{K}^i_{\mathcal{S}} = \{k_1, k_2, ..., k_{t'}\}$ *with* $t' = \binom{n-1}{s-1}$ *denote the key set received by each slave* $\mathcal{S}$*.*
2. $\mathsf{RecMes}(id_{node}, id_{win}, c, m)$ *on which master* $\mathcal{M}$ *receives a data frame containing message* m *from slave* $\mathcal{S}$ *checks if the counter is up-to-date then stores the packet in a queue of messages to be authenticated.*
3. $\mathsf{RecTag}(id_{node}, id_{win}, c, \mathsf{M\text{-}MAC}_{\mathbb{K}^i_{\mathcal{S}}}(id_{node}, id_{win}, c, m))$ *on which master* $\mathcal{M}$ *receives an authentication frame containing tag* $\mathsf{M\text{-}MAC}_{\mathbb{K}^i_{\mathcal{S}}}(id_{node}, id_{win}, c, m)$ *from slave* $\mathcal{S}$*. Further, he retrieves the packet matching the identifiers and counter from the queue and if the message proves to be authentic, i.e.,* $1 \leftarrow \mathsf{Ver}((id_{node}, id_{win}, c, m), k, \mathsf{M\text{-}MAC}_{\mathbb{K}^i_{\mathcal{S}}}(id_{node}, id_{win}, c, m))$*,* $\forall k \in \mathbb{K}^i_{\mathcal{S}}$*, he proceeds to authenticating the tag to other nodes with* $\mathsf{SendTag}(id_{node}, id_{win}, m, \mathbb{K}^i)$*. If the message is not available in the queue then the tag is discarded (subsequently an error message can be sent).*
4. $\mathsf{SendTag}(id_{node}, id_{win}, m, \mathbb{K}^i)$ *on which master* $\mathcal{M}$ *after receiving a message and its valid tag, groups all the remaining keys* $\mathbb{K}_{\mathcal{M}} \setminus \mathbb{K}^i_{\mathcal{S}}$ *in sets of size* v *then for each such set* $\widetilde{\mathbb{K}}^j_{\mathcal{S}}$ *computes* $\mathsf{M\text{-}MAC}_{\widetilde{\mathbb{K}}^j_{\mathcal{S}}}(id_{node}, j, m)$ *and broadcasts it in authentication frames with node identifier* id_{node} *and window identifier set to* j *(obviously there are* $|\mathbb{K}_{\mathcal{M}} \setminus \mathbb{K}^i_{\mathcal{S}}|/v$ *windows).*

and for each of the slaves $\mathcal{S}^*$:

1. $\mathsf{Setup}(\ell, n, s)$ *on which slave $\mathcal{S}_i$ obtains its key set $\mathbb{K}_{\mathcal{S}}^i = \{k_1, k_2, ..., k_{t'}\}$ with $t' = \binom{n-1}{s-1}$ from master $\mathcal{M}$ (either offline or via a secure channel).*
2. $\mathsf{RecMes}(id_{node}, id_{win}, c, m)$ *on which slave $\mathcal{S}$ receives a data frame containing message m from another slave $\mathcal{S}_j$ and proceeds similarly to master $\mathcal{M}$ by storing it in a queue of messages to be authenticated.*
3. $\mathsf{RecTag}(id_{node}, id_{win}, c, \mathsf{M\text{-}MAC}_{\mathbb{K}_{\mathcal{S}}^j}(id_{node}, id_{win}, c, m))$ *on which $\mathcal{S}_i$ receives an authentication frame containing tag $\mathsf{M\text{-}MAC}_{\mathbb{K}_{\mathcal{S}}^i}(id_{node}, id_{win}, c, m))$ from the master $\mathcal{M}$ or another slave $\mathcal{S}_j$ and verifies for all keys $k \in \mathbb{K}^i \cap \mathbb{K}^j$ if the tag is correct. If for all keys in its keyset a correct tag was received*
then message m is deemed authentic.
4. $\mathsf{SendMes}(m, \mathbb{K}_i)$ *on which slave $\mathcal{S}_i$ whenever wants to broadcast a message m increments its local counter, computes the tag $\mathsf{M\text{-}MAC}_{\mathbb{K}_{\mathcal{S}}^i}(id_{node}, 0, c, m)$ with its keyset $\mathbb{K}^i$ and sends the data frame containing m and an authentication frame containing the tag on the bus (note that in the case of slaves id_{win} is set to 0).*

Example 1. The key allocation done by the Setup procedure allows the keys to be split between groups of n slaves. Here we clarify our intentions with the *key splitting* procedure by giving an example. Table 1 shows the groups that can be formed in the case of 4 nodes. If we consider groups formed by exactly 2 nodes we have $\binom{4}{2} = 6$ groups and each two nodes share exactly $\binom{2}{0} = 1$ group. Table 1 outlines the groups shared by $\mathcal{S}_1$, i.e., G_9, G_{10}, G_{12}, and those shared by $\mathcal{S}_2$, i.e., G_5, G_6, G_{12}. Note that they intersect in one group G_{12}. In Table 2 the case of $n = 4$ and $n = 8$ nodes are explored, with complete groups of all sizes k and any number of corrupted nodes l. The total number of groups and the subgroup shared by each node as well as the percentage of secure bits, i.e., bits that cannot be forged by an adversary, from each $\mathsf{M\text{-}MAC}$ are outlined. Indeed, the percent of authenticated bits from each tag is higher and decreases significantly with the number of corrupted nodes.

Table 1. Possible groups with 4 nodes, groups of size 2 outlined in gray

	G_0	G_1	G_2	G_3	G_4	G_5	G_6	G_7	G_8	G_9	G_{10}	G_{11}	G_{12}	G_{13}	G_{14}	G_{15}
$\mathcal{S}_1$	0	0	0	0	0	0	0	0	1	①	①	1	①	1	1	1
$\mathcal{S}_2$	0	0	0	0	1	$\boxed{1}$	$\boxed{1}$	1	0	0	0	0	$\boxed{1}$	1	1	1
$\mathcal{S}_3$	0	0	1	1	0	0	1	1	0	0	1	1	0	0	1	1
$\mathcal{S}_4$	0	1	0	1	0	1	0	1	0	1	0	1	0	1	0	1

2.4 Variations of the Main Scheme: Two-Stage and Cumulative Authentication

For practical reasons we discuss two variations of the main scheme. In the experimental results section, the first variation is shown to have certain advantages in front of the main scheme for scenarios when nodes have equal computational power.

Table 2. Authentication rate in the case of $n = 4, 8$ participants, groups of size k and l corrupted nodes

				Authentication bits from one M-MAC (%)							
n	k	groups	sub-groups	$l = 0$	$l = 1$	$l = 2$	$l = 3$	$l = 4$	$l = 5$	$l = 6$	$l = 7$
4	1	4	1	25	25	25	25	-	-	-	-
4	2	6	3	50	33	33	16	-	-	-	-
4	3	3		75	25	0	0	-	-	-	-
8	1	8	1	12.5	12.5	12.5	12.5	12.5	12.5	12.5	12.5
8	2	28	7	25	21	17	14	10	7	3.5	0
8	3	56	21	37.5	26	17	10	5	1.7	0	0
8	4	70	35	50	28	14	5	1.4	0	0	0
8	5	56	35	62	26	8.9	1.7	0	0	0	0
8	6	28	21	75	21	3.5	0	0	0	0	0
8	7	8	7	87	12.5	0	0	0	0	0	0

In the case of *two-stage authentication* we assume a scenario in which only slave nodes are present, i.e., nodes with equal computational power. In this case each node can start broadcasting by sending a tag which includes only a part of the keys for the subgroups that he is part of and a second slave (pointed out by some flag, or predefined in protocol actions) continues with the authentication. The procedure is repeated until the desired number of authentication frames is reached. Various ways for tag allocation can be imagined. Consider the case of 8 nodes in subgroups of size 3 and 4 authentication frames (codenamed TS-8S3F4). If M-MACs are used then these can be set up to work in $GF(2^{16})$ or $GF(2^{32})$. Subsequently each node sends an M-MAC with keys for 4 of the nodes (or 2 in case $GF(2^{32})$) and the nodes reply in a round-robin fashion (note that a frame carries at most 64 bits). To save some computational power and have even more flexibility in tag allocation it is also possible to skip the use of the M-MAC. In Table 3 we give an example for this case. Each row corresponds to one of the 8 slaves and each column to one of the 56 groups that are formed with 3 slaves, $\times$ is used as placeholder to denote that a node is part of a group. Here f^i_j denotes the j-th part of frame i and the authentication is started by slave $\mathcal{S}_1$ with frame f^1_* followed by $\mathcal{S}_2$ with f^2_* then again $\mathcal{S}_1$ with f^3_* but this time followed by $\mathcal{S}_3$ with f^4_* (here $*$ is a placeholder for any of the frame components). We can set the size of each tag in f^2_* and f^2_* to 16 bits and for f^1_* and f^3_* use around 5-7 bits for each tag. This will result in a security level of around 64 bits for each node.

Since in some scenarios small delays may be acceptable, we can take benefit of them and increase the efficiency of the main scheme. In the *cumulative authentication* scheme a timer can be used and all messages are accumulated by the master over a predefined period δ then authenticated at once (this procedure can be employed in the slave-only settings as well). While this introduces an additional delay δ, similar to the case of the TESLA protocol, this delay can be chosen as small as needed to cover application requirements. Different to the case of the delay from TESLA like protocols, this delay is not strongly constrained by external parameters (such as oscillator precision, synchronization error, bus speed, etc.).

Table 3. Example of tag scheduling with two-stage authentication TS-8S2F4 (8 nodes with groups of size 3)

	G_1	G_2	G_3	G_4	G_5	G_6	G_7	G_8	G_9	G_{10}	G_{11}	G_{12}	G_{13}	G_{14}	G_{15}	G_{16}	G_{17}	G_{18}	G_{19}	G_{20}	G_{21}	G_{22}	G_{23}	G_{24}	G_{25}	G_{26}	G_{27}	G_{28}
S_1	f^1_1	f^1_2	f^1_3	f^1_4	f^1_5	f^1_6	f^1_7	f^1_8	f^1_9	f^1_{10}	f^3_1	f^3_2	f^3_3	f^3_4	f^3_5	f^3_6	f^3_7	f^3_8	f^3_9	f^3_{10}	f^3_{11}							
S_2	×	×	×	×	×	×																f^2_1	f^2_2	f^2_3	f^2_4	×	×	×
S_3	×						×	×	×	×	×											×	×	×	×	×		
S_4		×					×					×	×	×	×							×					×	×
S_5			×					×				×				×	×	×					×				×	
S_6				×					×				×			×			×	×				×				×
S_7					×					×				×			×		×		×				×			
S_8						×					×				×			×		×	×					×		

	G_{29}	G_{30}	G_{31}	G_{32}	G_{33}	G_{34}	G_{35}	G_{36}	G_{37}	G_{38}	G_{39}	G_{40}	G_{41}	G_{42}	G_{43}	G_{44}	G_{45}	G_{46}	G_{47}	G_{48}	G_{49}	G_{50}	G_{51}	G_{52}	G_{53}	G_{54}	G_{55}	G_{56}
S_1																												
S_2	×	×	×	×	×	×	×	×																				
S_3									f^4_1	f^4_2	f^4_3	f^4_4	×	×	×	×	×	×										
S_4	×	×							×	×	×	×							×	×	×	×	×	×				
S_5			×	×	×				×				×	×	×				×	×	×				×	×	×	
S_6			×			×	×			×			×			×	×		×			×	×		×	×		×
S_7	×			×		×		×			×			×		×		×		×		×		×	×		×	×
S_8		×			×		×	×				×			×		×	×		×	×		×	×		×	×	×

2.5 Increasing Security with LM-MACs (Linearly Mixed MACs)

As outlined in our abstract description, M-MACs use an array of keys to build a tag which is verifiable by any of the keys. The first security property which we require for an M-MAC is *unforgeability* and is a standard property for any MAC code, thus it merely derives from the main building block. We do develop on this by requiring a new property which we call *strong non-malleability* and which we show to be achievable by our more advanced LM-MAC construction.

Construction 3. *(Linearly Mixed MAC) We define the* LM-MAC *as the tuple of probabilistic polynomial-time algorithms* (Gen, Tag, Ver) *that work as follow:*

1. $\mathbb{K} \leftarrow \mathsf{Gen}(1^\ell, s)$ *is the key generation algorithm which flips coins and returns a key set* $\mathbb{K} = \{k_0, k_1, ..., k_s\}$ *where each key has* ℓ *bits (*ℓ *is the security parameter of the scheme),*
2. $\tau \leftarrow \mathsf{Tag}(\mathbb{K}, \mathbb{M})$ *is the mac generation algorithm which returns a tag* $\tau = \{x_1, x_2, ..., x_s\}$ *where each* x_i *is the solution of the following linear system in* $GF(2^b)$*:*

$$\begin{cases} \mathsf{KD}_1(k_1, m_1) \cdot x_1 + \mathsf{KD}_2(k_1, m_1) \cdot x_2 + ... + \mathsf{KD}_s(k_1, m_1) \cdot x_s \equiv \mathsf{MAC}_{k_1}(m_1) \\ \mathsf{KD}_1(k_2, m_2) \cdot x_1 + \mathsf{KD}_2(k_2, m_2) \cdot x_2 + ... + \mathsf{KD}_s(k_2, m_2) \cdot x_s \equiv \mathsf{MAC}_{k_2}(m_2) \\ ... \\ \mathsf{KD}_1(k_s, m_s) \cdot x_1 + \mathsf{KD}_2(k_s, m_s) \cdot x_2 + ... + \mathsf{KD}_s(k_s, m_s) \cdot x_s \equiv \mathsf{MAC}_{k_s}(m_s) \end{cases}$$

Here b *is polynomial in the security parameter* ℓ *and* KD *stands for a key derivation process. If such a solution does not exist, then the* M-MAC *algorithm fails and returns* $\perp$.
3. $v \leftarrow \mathsf{Ver}(k, m, \tau)$ *is the verification algorithm which returns* 1 *if and only if having* $\tau' = \mathsf{MAC}_k(m)$ *it holds* $\tau' \equiv \mathsf{KD}_1(k, m) \cdot x_1 + \mathsf{KD}_2(k, m) \cdot x_2 + ... + \mathsf{KD}_s(k, m) \cdot x_s$. *Otherwise it returns* 0.

Let us emphasize that the probability that the M-MAC fails to return a solution is negligible in the security parameter (if proper b and s are chosen). As shown in [3] the probability that an n by n matrix with random elements from $GF(q)$ is non-singular converges to $\prod_{i=1}^{\infty}(1 - 1/q^i)$ as $n \to \infty$. For example in case when $s = 4$ we have a chance of around 10^{-5} for $b = 16$ and 10^{-10} for $b = 32$ for the M-MAC to fail.

Example 2. We want to clarify here our intentions on M-MACs with respect to the protocol design. Consider a case when master M broadcasts messages m_1 and m_2 to slaves S_1, S_2 along with the authentication tag. To increase efficiency of our protocol we want to authenticate both messages with the same mixed MAC and more, since only a portion of each tag is disclosed (reducing the bus overhead but also the security level), we want one of the slaves to be able to carry out the authentication further with a new part of a valid tag (note that this is what happens in the case of the two-stage authentication). Consider that the following packets arrive on the bus: message m_1, message m_2 and the mixed tag obtained by simply concatenating the two tags $\mathsf{MAC}_{k_1}(m_1)||\mathsf{MAC}_{k_2}(m_2)$. However, due to the message filtering of the CAN bus it may be that the two messages do not reach both slaves. Assume message m_1 reaches S_1 and m_2 reaches S_2. Now neither S_1 or S_2 can carry the authentication further, even in the case when they both have k_1 and k_2 they are not in possession of the message that reached the other slave and thus they can not validate the other part of the tag. More relevant, note that the nodes are unable to detect if the other part of the tag is compromised. Now consider the case of the LM-MAC. In this case the tag is obtained by mixing the two tags via the linear equation system, e.g., the two components of the tag x_1, x_2 verify a relation of the form $\alpha_1 x_1 + \alpha_2 x_2 = \mathsf{MAC}_{k_1}(m_1)$ and $\beta_1 x_1 + \beta_2 x_2 = \mathsf{MAC}_{k_2}(m_2)$ (here α's and β's are derived from the secret keys k_1, k_2). If an adversary compromises any part of the tag, i.e., either x_1 or x_2, then both equations will fail to verify and any of the receivers can detect this (indeed, we assume that the adversary is not in possession of the secret keys k_1 and k_2 since in such case he can compute correct LM-MACs anyway). Consequently, with the LM-MACs any of them can check the tag for correctness and this validation will also hold for the other receiver, this is inherited from the strong non-malleability property for M-MACs.

We now sketch a more formal account of the properties that we require for our building blocks. These are mediated by two attack games against unforgeability, i.e., $\mathsf{Game}^{\mathsf{UF}}_{\mathsf{M\text{-}MAC}}$, and strong non-malleability, i.e., $\mathsf{Game}^{\mathsf{SNM}}_{\mathsf{M\text{-}MAC}}$. Both games are defined for a generic M-MAC construction and in particular the LM-MAC can be proved to resist such attacks. The attack game on strong non-malleability $\mathsf{Game}^{\mathsf{SNM}}_{\mathsf{M\text{-}MAC}}$ against an M-MAC requires an adversary to be able to construct an M-MAC in such way that verification fails with at least one of the keys but succeeds with another. An M-MAC that is resilient to such an attack is called *strongly non-malleable*.

Definition 1. *(Unforgeability Attack Game) We define the* M-MAC *unforgeability game* $\mathsf{Game}^{\mathsf{UF}}_{\mathsf{M\text{-}MAC}}$ *as the following five stage game between challenger $\mathcal{C}$ and adversary Adv:*

1. Challenger $\mathcal{C}$ runs the key generation algorithm $\mathsf{Gen}(1^{\ell}, s)$ *to get a key set* $\mathbb{K} = \{k_0, k_1, ..., k_s\}$.
2. Adversary Adv is allowed to requests $\mathcal{C}$ any subset of the keyset $\mathbb{K}' = \{k_{j_0}, k_{j_1}, ..., k_{j_t}\}$, $t < s$ *where* $\forall j_i \in [1..s]$. *That is, the adversary is always missing at least 1 of the keys.*

3. Adversary Adv is allowed to make queries to the MAC generation oracle $\mathcal{O}^{\mathsf{Tag}}(\mathbb{K}, \mathbb{M})$ *for any message tuple* $\mathbb{M}$ *to obtain the corresponding tag* $\tau \leftarrow \mathsf{Tag}(\mathbb{K}, \mathbb{M})$ *and to the verification oracle* $\mathcal{O}^{\mathsf{Ver}}(i, \tau, m)$ *with any key index* i*, tag* τ *and message* m *and the oracle will return* 1 *if and only if* τ *is a correct tag under key* k_i *for message* m*.*
4. Eventually, the adversary outputs the tuple $(m^{\diamondsuit}, \tau^{\diamondsuit}, i)$ *for some index* i *such that he is not in possession of* k_i*.*
5. The game output is 1 *if the following two conditions hold:* Ver *outputs* 1 *on* (τ, m, k_i) *and the adversary never queried* m *to the* Tag *oracle. Otherwise the game output is* 0.

Definition 2. *(Unforgeability) We say that a mixed message authentication code* $\mathsf{M\text{-}MAC}$ *is unforgeable if:* $\Pr\left[\mathsf{Game}^{\mathsf{UF}}_{\mathsf{M\text{-}MAC}}(1^{\ell}, s) = 1\right] < \mathrm{negl}(\ell)$.

Definition 3. *(Strong Non-malleability Attack Game) We define the* $\mathsf{M\text{-}MAC}$ *strong non-malleability game* $\mathsf{Game}^{\mathsf{SNM}}_{\mathsf{M\text{-}MAC}}$ *as the following five stage game between challenger* $\mathcal{C}$ *and adversary* Adv*:*

1. Challenger $\mathcal{C}$ *runs the key generation algorithm* $\mathsf{Gen}(1^{\ell}, s)$ *to get a key set* $\mathbb{K} = \{k_0, k_1, ..., k_s\}$*.*
2. Adversary Adv *is allowed to requests* $\mathcal{C}$ *any subset of the keyset* $\mathbb{K}' = \{k_{j_0}, k_{j_1}, ..., k_{j_t}\}$*,* $t < s - 1$ *where* $\forall j_i \in [1..s]$*. That is, the adversary is always missing at least 2 of the keys.*
3. Adversary Adv *is allowed to make queries to the MAC generation oracle* $\mathcal{O}^{\mathsf{Tag}}(\mathbb{K}, \mathbb{M})$ *for any message tuple* $\mathbb{M}$ *to obtain the corresponding tag* $\tau \leftarrow \mathsf{Tag}(\mathbb{K}, \mathbb{M})$ *and to the verification oracle* $\mathcal{O}^{\mathsf{Ver}}(i, \tau, m)$ *with any key index* i*, tag* τ *and message* m *and the oracle will return* 1 *if and only if* τ *is correct tag under key* k_i *for message* m*.*
4. Eventually, the adversary outputs the pair $(m^{\diamondsuit}, \tau^{\diamondsuit})$*.*
5. The game output is 1 *if there are at least two keys* $k, k' \in \mathbb{K}$ *such that the following two conditions hold:* Ver *outputs* 1 *on* (τ, m, k) *but outputs* 0 *on* (τ, m, k') *and the keys* k, k' *are not part of the adversary keyset* $\mathbb{K}'$*. Otherwise the game output is* 0.

Definition 4. *(Strong Non-malleability) We say that a mixed message authentication code* $\mathsf{M\text{-}MAC}$ *is strongly non-malleable if:* $\Pr\left[\mathsf{Game}^{\mathsf{SNM}}_{\mathsf{M\text{-}MAC}}(1^{\ell}, s) = 1\right] < \mathrm{negl}(\ell)$.

Theorem 1. *The* $\mathsf{LM\text{-}MAC}$ *construction is unforgeable if the underlying MAC is unforgeable and is strongly non-malleable in the random oracle model.*

Due to space limitations, a proof of this theorem in the random oracle model is deferred for the extended version of this work.

3 Experimental Results

To evaluate the performance of the proposed protocol suite, we used several setups with different hardware components to determine the minimum authentication delay. Automotive grade embedded devices from Freescale and Infineon as well as a notebook equipped with an adapter for CAN communication from Vector were employed to build the nodes of our experimental CAN network. The embedded platforms that we used are representatives for industry's low-end and high-end edges.

3.1 Test Beds

Using the aforementioned components we built several test beds. First, the case of a system using the centralized authentication approach with one master node and 4 slave nodes was considered:

- *Testbed 1:* $S12 + 4 \times S12$. Both master and slave nodes are built on identical S12 development boards with CAN communication speed set to 125kbps.
- *Testbed 2:* $TC1782 + 4 \times TC1797$. Master and slave nodes are built on similar TriCore development boards having the same computational and communication capabilities. CAN communication speed is set to 1Mbps.
- *Testbed 3: Intel* $T7700 + 4 \times S12$. The master node is implemented on a PC (Intel Core2Duo CPU T7700@2.4GHz) while slave nodes are built on the S12 boards. The master-slave CAN communication is done through the CANcardXL using a low speed CANcab for 125kbps.
- *Testbed 4: Intel* $T7700 + 4 \times TC1797$. The master node is implemented on the same PC as in the previous case while slave nodes are built on the TriCore platform. This time a high speed CANcab is used with the CANcardXL to enable a 1Mbps communication speed.

A different testbed was set up to compare the different variants of the key splitting protocol on a system with 8 slaves based on S12X nodes. Two variants were considered as we further discus: centralized authentication (in this case one extra node was added to act as the master) and two-stage authentication.

3.2 Protocol Performance

Centralized authentication was implemented on the four testbeds prepared for this purpose. Our implementation considers 6 groups of two nodes each formed by combining the four available nodes. Messages and authentication tags are always sent as separate frames and the message size is always 8 bits. The MAC size for each group is set to 21 bits so that 3 authentication tags fit a single 64 bit CAN frame. The MAC is computed using the MD5 hash function over an input formed by concatenating the group key to the message. The resulting hash is then truncated to the desired size. Table 4 holds the timings and bus loads for each test bed. Here δ is the authentication delay, i.e., the time needed by a node to authenticate the message once it receives it. For the bus load we considered the fraction of traffic caused by the authentication tags over the entire bandwidth.

As expected, scenarios in which high end devices played the role of master nodes (PC, TriCore) showed better performance than in the case of low end master nodes. The case of a PC master with TriCore slaves does not perform better, despite the considerable difference in computational power between master nodes (TriCore vs. Intel Core2Duo) due to limitations of CAN adapters. Because of their internal hardware/software design, these adapters introduce some limitations, e.g., the average response time specified by Vector for the CANcardXL is $100\mu s$.

To evaluate the protocol behavior when using different trade-offs we implemented different variants of the key splitting authentication protocol on a system with 8 slaves built on S12X nodes. By grouping the eight nodes two by two we obtain a total of 28 groups. The size of the authentication tags and the truncated MAC size differ in each variant. We set up the implementations as follows:

- *Centralized:* The message sending node computes and sends one MAC for each group that he is part of. The master computes and sends one MAC for each of the other 21 groups (if groups of size 2 are used). If the master is to perform the authentication in only 2 frames then each MAC can be truncated to 5 bits and this will lead to a total of 35 security bits for each node. But if we increase the number of authentication frames from the master to 3, then each MAC can be truncated to 9 bits giving a total of 63 authentication bits for each node which is a reasonable level for real-time security.
- *Two-stage:* The master node is missing in this implementation, therefore we use two helper nodes for computing and sending the complete authentication tag. In the two-stage variant, the sender node will first put one authentication tag on the bus which contains the full 36 authentication bits for one of the helper nodes, 20 bits for the second one and 8 extra bits for another node. This first tag is followed by a second tag generated by the first helper node which contains the remaining 16 authentication bits for the second helper node and 48 bits equally distributed for three of the remaining nodes. To complete the 36 authentication bits for each of the remaining nodes, the sender node and the second helper node will each put an authentication tag on the bus. As discussed previously, the security level can be raised to around 64 bits by using groups of size 3 and the described tag allocation procedure.

Table 5 holds the results achieved with these two implementations. The worst performer in terms of authentication delay is the implementation of the centralized authentication variant as it involves computing MACs for each of the 28 groups in a sequential manner. In the other implementation, a smaller number of MACs are computed some of which are done by different nodes in parallel. A smaller authentication delay is obtained when using the two-stage implementation at the cost of an increased CPU load on the sender side. However, this cost is somewhat compensated by the higher level of security offered by the fact that the sender node offers more authentication bits.

3.3 Computational Performance with Linearly Mixed Tags

The results from Tables 4 and 5 use the simple concatenation of individual MACs computed with MD5 as the underlying hash function. We now take a brief account of the impact of mixing tags using linear systems of equations, complete experimental results on this will be available in the extended version of our work, here we make an accurate estimation of the computational costs. To begin with, in Table 6 we give an overview on the computational timings for various hash functions and input sizes on both of the employed platforms. For the Linearly Mixed MACs, in addition to the computation of the MACs, two supplemental computational tasks are required: solving the linear system of equations on the sender side (a task which should be usually done by the master which has higher computational power) and reconstructing the MAC on the receiver side. Our experimental results obtained on the communication master equipped with the Intel 2.4GHz core with the well known NTL library (http://www.shoup.net/ntl/) showed that the computational cost of solving the system for 2 nodes in $GF(2^8)$ up to $GF(2^{32})$ are

Table 4. Centralized authentication with 4 nodes

Master	Slave	δ	Bitrate	Bus load
S12X	4xS12X	2.54 ms	125 kbps	53.84%
PC	4xS12X	1.848 ms	125 kbps	72.22%
TriCore	4xTriCore	267 μs	1 Mbps	54.31%
PC	4xTriCore	378 μs	1 Mbps	42.54%

Table 5. Centralized & Cascade with 8 nodes

Variant	Master	Slave	δ	Bus load
Centralized	S12X	8xS12X	22.624 ms	11.27%
Two-stage	-	8xS12X	6.806 ms	46.21%

around 3–6 times more intensive than an MD5 computation and this increases to 10–20 times the MD5 computation in the case of 4 nodes. Since this task should be done by the master node it shouldn't raise computational issues. The reconstruction of the MAC was around 10 times cheaper compared to the linear mixing procedure and compared to MD5 it was in the range of 0.5–5 times more intensive, the later in the case of 8 nodes and $GF(2^{32})$. All these are reasonable amounts of computations and we believe that they can be significantly improved with platform dependent tweaks.

Table 6. Computational performance of employed embedded platforms

Hash function	Input size (bytes)					
	S12			TriCore		
	0	16	64	0	16	64
MD5	$371\mu s$	$374\mu s$	$1414\mu s$	$10.16\mu s$	$11.00\mu s$	$18.34\mu s$
SHA1	$1.144ms$	$1.148ms$	$4.510ms$	$14.64\mu s$	$15.10\mu s$	$27.60\mu s$
SHA256	$2.755ms$	$2.755ms$	$5.440ms$	$41.70\mu s$	$42.35\mu s$	$80.80\mu s$

4 Backward Compatibility with CAN+

There are two main drawbacks to the LiBrA-CAN protocol. First of all, depending on the setup, it can require quite a lot of the CAN bus' bandwidth, and second, all nodes in the system need to be aware of the LiBrA-CAN protocol for it to work. In this section, we show a method of eliminating both of these drawbacks using an unofficial extension of the CAN protocol, called CAN+ [14].

The CAN+ protocol allows transmission of extra data along with a CAN packet on an out-of-band channel. It does this by transmitting data at an increased rate in between CAN sample points. At least 225 extra bits can be transmitted with the CAN+ protocol alongside a CAN message.

Using the CAN+ protocol for LiBrA-CAN data transmission helps in two ways. First of all, the required bandwidth drops. For LiBrA-CAN schemes whereby a single node never needs to transmit more than $\lceil \frac{225}{64} \rceil = 3$ authentication tags, all LiBrA-CAN data can be transmitted as CAN+ data. This reduces the LiBrA-CAN overhead for those schemes to 0%. Nodes that need to transmit just a tag, can do so by transmitting a 0-byte CAN message and embedding the tag as CAN+ data, thereby reducing the time they use the bus from 108 bit lengths (for an 8-byte message) to 44 bit lengths in non-extended CAN mode, which is a 60% decrease.

Second, if LiBrA-CAN authentication data is only transmitted as CAN+ data, then nodes that do not support CAN+ will not even see the LiBrA-CAN data. Thus, a system can be setup whereby important nodes are outfitted with a CAN+ transceiver, while non-important nodes aren't. This makes the LiBrA-CAN protocol completely backwards compatible with existing CAN networks: nodes supporting CAN+ could be dropped into the network at will and start authentication messages with LiBrA-CAN, while existing CAN nodes will be completely oblivious as to what is going on and continue functioning as before. An added bonus is that this also drastically reduces roll-out cost.

5 Discussion and Conclusions

The proposed protocol is efficient when the number of nodes is low. We expect this to be the case in many automotive scenarios where, although the number of ECUs may be high, the numbers of manufacturers from which they come may not be high and distributing trust between several groups is an acceptable solution. If the number of nodes is too high we see only two resolutions: public key cryptography (with the drawback of high computational requirements, at least 2 orders of magnitude) or TESLA like protocols (with the drawback of authentication delays as shown in [6]). CANAuth [12] is also a solution for high number of nodes if one considers that source authentication is not relevant and associating keys to message groups is sound from a security perspective. While a decision on what protocol should be used for in-vehicle authentication can be taken only by manufacturers and by means of consortium, we believe that this proposal should be considered as an interesting alternative. The use of MAC mixing, key splitting and the features of the CAN arbitration seems to give an efficient management for source authentication.

Acknowledgement. This work was supported by in part by research grants PNII-IDEI 940/2008 and POSDRU 107/1.5/S/77265, inside POSDRU Romania 2007-2013. It was also supported in part by the Research Council KU Leuven: GOA TENSE (GOA/11/007), by the Flemish IBBT projects, and by the European Commission through the ICT programme under contract ICT-2007-216676 ECRYPT II. In addition, it was supported by the Flemish Government, FWO G.0550.12N and by the European Commission through the ICT programme under contract FP7-ICT-2011-284833 PUFFIN. We also thank the reviewers for helpful comments on our work.

References

1. Bar-El, H.: Intra-vehicle information security framework. In: Proceedings of 9th Embedded Security in Cars Conference, ESCAR (2009)
2. Chan, H., Perrig, A., Song, D.X.: Random key predistribution schemes for sensor networks. In: 2003 Proceedings of the Symposium on Security and Privacy, pp. 197–213. IEEE (2003)
3. Charlap, L.S., Rees, H.D., Robbins, D.P.: The asymptotic probability that a random biased matrix is invertible. Discrete Mathematics 82(2), 153–163 (1990)

4. Checkoway, S., McCoy, D., Kantor, B., Anderson, D., Shacham, H., Savage, S., Koscher, K., Czeskis, A., Roesner, F., Kohno, T.: Comprehensive experimental analyses of automotive attack surfaces. In: USENIX Security 2011 (2011)
5. Fiat, A., Naor, M.: Broadcast Encryption. In: Stinson, D.R. (ed.) CRYPTO 1993. LNCS, vol. 773, pp. 480–491. Springer, Heidelberg (1994)
6. Groza, B., Murvay, P.-S.: Higher layer authentication for broadcast in Controller Area Networks. In: International Conference on Security and Cryptography, SECRYPT (2011)
7. Koscher, K., Czeskis, A., Roesner, F., Patel, S., Kohno, T., Checkoway, S., McCoy, D., Kantor, B., Anderson, D., Shacham, H., Savage, S.: Experimental security analysis of a modern automobile. In: 2010 IEEE Symposium on Security and Privacy, SP, pp. 447–462 (May 2010)
8. Naor, M., Pinkas, B.: Threshold Traitor Tracing. In: Krawczyk, H. (ed.) CRYPTO 1998. LNCS, vol. 1462, pp. 502–517. Springer, Heidelberg (1998)
9. Perrig, A., Canetti, R., Song, D.X., Tygar, J.D.: SPINS: Security protocols for sensor networks. In: Seventh Annual ACM International Conference on Mobile Computing and Networks, MobiCom 2001, pp. 189–199 (2001)
10. Perrig, A., Canetti, R., Tygar, J.D., Song, D.X.: Efficient authentication and signing of multicast streams over lossy channels. In: IEEE Symposium on Security and Privacy, pp. 56–73 (2000)
11. Roeder, T., Pass, R., Schneider, F.: Multi-verifier signatures. Journal of Cryptology 25(2), 310–348 (2012)
12. Van Herrewege, A., Singelee, D., Verbauwhede, I.: CANAuth-a simple, backward compatible broadcast authentication protocol for CAN bus. In: 9th Embedded Security in Cars Conference (2011)
13. Wolf, M., Weimerskirch, A., Paar, C.: Secure in-vehicle communication. In: Embedded Security in Cars, pp. 95–109 (2006)
14. Ziermann, T., Wildermann, S., Teich, J.: CAN+: A new backward-compatible Controller Area Network (CAN) protocol with up to 16x higher data rates. In: DATE, pp. 1088–1093. IEEE (2009)