LiBrA-CAN: Lightweight Broadcast Authentication for Controller Area Networks

Bogdan Groza, Stefan Murvay, Anthony Van Herrewege, Ingrid Verbauwhede

Abstract—Despite realistic concerns, security is still absent from vehicular buses (such as CAN) mostly due to technical challenges: low bandwidth and processing power, low cost margins, etc. Here we design an efficient 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 on each node, we share keys between groups of nodes which leads to a higher security level in case when compromised nodes form only a minority. Based on practical arguments, we recognize this assumption to be realistic for automotive networks. Subsequently, amalgamating regular message authentication codes with systems of linear equations increases the chances for a forgery to be detected. We present several protocol variants that are extremely flexible and set way for different trade-offs on bus load, computational cost and security level, taking into account the most recent developments such as the recently released CAN-FD standard. To gain full compatibility with existent networks, we also discuss a backward compatible solution based on CAN+, a recently proposed extension to CAN. Finally, we present experimental results on state-of-the-art Infineon TriCore controllers which are contrasted with low-end Freescale S12X cores, both are wide spread devices from the automotive industry.

1 Motivation and related work

Vehicular network security established itself as an intense research topic in the last few years. Outstanding experimental results from Koscher et al. [14] and later Checkoway et al. [7] showed vehicles to be easy targets for malicious adversaries. The myriad of attacks reported in the last five years showed that virtually any subsystem inside a car is vulnerable to attacks that exploit the CAN bus as an entry point. Table 1 summarizes some of the attacks reported so far, these are grouped according to the vehicular subsystem on which they act: power train, chassis, body, multimedia, telematics, HMI or active/passive safety. These attacks are reproducible on most (if not any) cars on the market. Any device inside a car can be seriously affected by real-world adversaries and this is no surprise as long as CAN (the bus used to connect all relevant communications inside a vehicle) lacks cryptographic security. The same holds for other communication buses inside the car, even for the most recent developments, e.g., FlexRay, CAN-FD.

B. Groza and S. Murvay are with the Faculty of Automatics and Computers, Politehnica University of Timisoara, Timisoara, Romania. Email: bogdan.groza@aut.upt.ro, stefan.murvay@gmail.com A. Van Herrewege and I. Verbauwhede are with ESAT/COSIC - iMinds, KU Leuven, Belgium. Email: anthony.vanherrewege@esat.kuleuven.be, ingrid.verbauwhede@esat.kuleuven.be

Proposals for assuring security for in-vehicle buses are not many and there are several clear reasons behind this. First, the relevance of assuring security inside vehicles was decisively proved only in the recent years [14], [7]. Second, the design principles used by manufacturers are somewhat out of reach for the academic community, making it hard to understand many assertions behind protocol designs. 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. Several approaches to in-vehicle security advocate the use of secure gateways between different ECUs (Electronic Control Unit) or subnetworks [1], [27] and rely on basic cryptographic constructions (encryptions, signatures, etc.). In particular, these are not meant specifically for assuring broadcast authentication on CAN which is still the most common communication bus in automotives; we discuss next such proposals.

TESLA and CANAuth. TESLA like protocols proved to be highly effective in sensor networks [18], [17] and so far are the most efficient alternative for assuring broadcast authentication with cheap Message Authentication Codes (MAC). However, when it comes to the 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 [9] is to determine a lower bound on these delays and establish some trade-offs. Delays in the order of milliseconds, as shown to be achievable in [9], are satisfactory for many scenarios, but such delays do not appear to be small enough for in-vehicle communication. There is no obvious way to improve on these delays. 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 [26] 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

TABLE 1
Recently reported attacks having CAN as entry-point

Subsystem	Affected module	Consequence
Power Train (longitudinal propulsion: engine, transmission, etc.)	Engine	Increase idle RPM, temporary RPM increase, initiate crankshaft re-learn (disturbs engine timings), disable cylinders, kill engine, grind starter, remote car start, cannot turn on (DoS to/from BCM), cannot turn off (while turned on, cause BCM to activate ignition output) [14]
Chassis (wheels and their relative position and movement: steering, braking, etc.)	Brakes	Disable, engage front left, engage front right, unlock front left, unevenly engage right brakes, releases brakes (prevents braking) [14]
	Power steering	Disable [14]
Body (entities that do not belong to vehicle dynamics: wipers, lighting, window lifter, air conditioning, seats, etc.)	Doors	Lock/unlock car, unlock all while at speed [14]
	Gauges & instruments	Falsify speedometer reading, falsify fuel level, freeze display panel [14]
	Electric window lift	Control windows [11], DoS [12], disable window relays [14]
Multimedia, Telematics and HMI (information exchange: display, switches, radio, navigation, Internet, etc.)	Radio	Increase volume, change display, produce ticking sound [14]
	Driver Information Center	Change display [14]
Active/Passive Safety (airbags, warnings, seatbelt, ABS, ESP, cruise control, etc.)	Airbag	Suppress missing airbag warnings by emulating its presence [12]

be so practical. For this purpose, in [26] a clever solution is imagined: the keys are linked to multiple ID codes using masks, which greatly reduces the number of keys. But still, this leads to some security concerns. 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 to on CAN. It is thus unclear which keys can be shared with different manufacturers and how or what the security outcomes of this are. Obviously, if a third party device is easy to compromise (even an innocuous one such as passive receiver) then all the IDs which it was allowed to authenticate are equally compromised.

Voting. Szilagyi and Koopman introduce a validity voting scheme in [23] and [24]. The scheme is intended for generic time-triggered communication such as TT-CAN, FlexRay, etc. The core part of the protocol relies on the classical paradigm of sharing keys between each sender and receiver then authenticating packets on a one MAC per receiver basis. Further, to make it feasible to embed the MACs in a single frame, the tags are truncated and concatenated (e.g., 3 MACs each of 8 bits are fitted at the end of a single frame). The communication is time triggered, each receiver releasing his message and his vote on previous messages in fixed time slots. Both the new message and his vote, along with all previously received messages, are authenticated under the same array of MACs to other receivers. The scheme appears to be a trade-off between computational time, authentication delays and bandwidth in order to fit the authentication bits in one frame. Indeed, if the frame would be larger, and the sender could fit more MAC bits in each frame, then authentication could be done at once within a single frame without needing to wait for the votes of the other nodes. This would improve both on delays (as nodes will not need to wait for the vote of other nodes) and computational power since, indeed, the nodes that subsequently vote are re-authenticating messages that were previously authenticated with a small amount of bits. The procedure leads to a drawback as stated in [24]: for frames that are lost, the receive history of the nodes does not match and authentication will fail for these frames. As suggested in [24] this can be fixed by adding additional bits for lost packets, but sufficient votes from other nodes would still be required to deem the frame authentic.

LibraCAN. Our protocol design is based on two paradigms: key splitting and MAC mixing, the later procedure is optional and is intended to increase security by allowing each node to detect a potential forgery. In addition to these, authentication can be achieved in a progressive manner by revealing only a few bits of the MAC in each packet to each verifier (this is mostly intended for the case of standard CAN frames that cannot accommodate more than 64 bits of data).

Key splitting allows a higher entropy for each mixed MAC that is sent at the cost of losing some security for groups that contain more malicious 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. Note that in contrast to the scheme of Szilagyi and Koopman which requires the nodes to be present and vote, LiBrA benefits from a majority of non-corrupted nodes, but does not require their presence to vote (it is just their keys that need to be safe). This procedure is not new, similar techniques were proposed in the past in the context of broadcast security. We could trace this back up to the work of Fiat and Naor [8], but there is a large amount of papers on this subject. The work of Canetti et al. [3] provides efficient constructions based on the same principles. 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 [5], pay-TV [16], etc. The main idea behind such schemes is that groups of l corrupted receivers cannot learn the secret (in settings with $n > l$ users). One interesting feature of such protocols, which we consider relevant for the setting here, is the ability to trace corrupted keys [16]. While this feature is not directly exploited in our protocol, it can be used in our setting as well to detect malicious nodes (roughly, a corrupted receiver has some chances in forging a MAC but the probability that his forgery is detected increases exponentially with the number of forgeries).

In addition to this, we exhibit a distinct contribution in the construction of Linearly Mixed MACs which allows us to

TABLE 2
Specifications, advantages and limitations of current proposals for assuring CAN security

Protocol	Specifications and advantages	Possible Limitations
TESLA-CAN [9]	- efficient use of symmetric primitives with time synchronization - successful and well studied in sensor networks	- fixed authentication delays (usually milliseconds), - time triggered release of keys (potential conflicts with CAN arbitration) - resynchronization can be an issue at very small authentication delays
CAN-Auth [26]	- ID oriented authentication, follows in detail CAN specifications - no authentication delays or time synchronization	- no source authentication (unclear security implications, e.g., third party tools can inject forged frames) - number of keys increases with number of IDs
Voting [10]	- shared symmetric keys between each 2 nodes - each node votes for the authenticity of previous messages	- small number of nodes - time triggered release of authentication tags (nodes need to wait over multiple time slots to get sufficient votes) - nodes are required to be present and vote - disagreements between nodes if packets are lost - each node needs to apply a MAC on his current message, his votes on previous messages, and all messages that he received previously
Libra-CAN	- efficient source authentication with dishonest minority - efficient forgery detection with MAC mixing - no authentication delays or time synchronization	- small number of participants - malicious nodes in minority

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 case benign nodes are in possession of a wrong key. To the 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. [21]. 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. For our construction we require that the mixed MACs are strongly non-malleable, a property which appears to be entirely different.

For our setting we assume a reduced number of participants. To strengthen our assumption we present in Figure 1 the network topology of a high-end vehicle based on [15]. Indeed, it is easy to see that while there is a high number of ECUs, not all of them share the same network, and consequently they can be easily placed into small broadcast groups based on the subnetwork they are part of. 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 there appears to be little or no benefit for this. 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.

It is also an advantage for the current proposal that this year a new CAN standard that supports flexible data rates was released: CAN-FD [20]. This is strong evidence that CAN will persist in the industry despite more recent developments such as FlexRay. The larger data frames of CAN-FD [20] allow us to design a more robust authentication protocol that significantly reduces the overhead of independent authentication frames, we detail this in the main version of the protocol. Besides the larger data field, CAN-FD allows an increased data rate after the arbitration which will make it even faster to carry the authentication data. We present real-time simulation

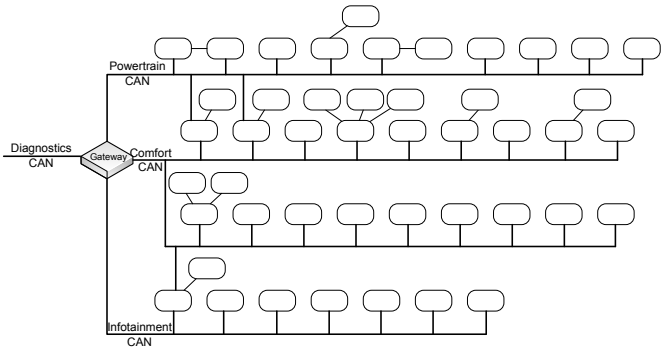

Fig. 1. Network topology of a VW Phaeton based on [15]

of our protocol using the industry standard CANoe tool from Vector (www.vector.com).

2 The Protocol

We begin with a brief overview of the application setting and assumptions. Then we outline the main authentication scheme and discuss some variations or improvements to it.

2.1 Setting, assumptions and goals

The Controller Area Network (CAN) is a broadcast serial bus designed by Bosch [19], [13]. The typical topology consists of a differential bus which connects multiple nodes by two wires (called CAN-High and CAN-Low). This topology is also suggested in Figure 4.

In this setting, we do assume the usual presence of a Dolev-Yao adversary that has full control over the communication channel. That is, he can eavesdrop, modify and send messages at his will. Of course, the goal of our scheme is to assure an authentic channel, i.e., to prevent messages that originate from the adversary to be accepted by the honest principals. This is achieved by authentication tags added to each message or separately sent (according to the protocol variant). Further, the security and efficiency gain stem from the way in which we share keys between the nodes (avoiding the simple but less efficient pair-wise sharing) and the way in which we build

the authentication tags (while a simple concatenation of the authentication tags can be also employed in our scheme, we view this rather as a basic approach and propose the more elegant linear mixing).

2.2 Frame structure

CAN frames carry at most 8 bytes of data. Each CAN frame begins with a start bit followed by the arbitration field (29 bits in extended frames and 11 bits in standard frames), a control field (6 bits), data bits (0-64), CRC sequence (15 bits), a 2 bit acknowledgement and 7 bits that mark the end of frame. Stuffing bits are added after each 5 consecutive bits of identical value. The newer CAN-FD standard [20] allows up to 64 bytes of data to be carried by one frame and more, the data rate can be increased after the arbitration procedure.

For the case of standard CAN frames (unable to carry both the data and authentication tag), as well as for some variations of the main scheme, we separate between message frames and authentication frames. Larger data blocks or authentication tags (exceeding 8 bytes) can be split across multiple frames with the same ID field and counter. On the other side, with CAN-FD frames, it is advantageous to embed the authentication tag in the message frame and take benefit of the increased data rate that follows the arbitration procedure. This also reduces the authentication delay and allows immediate verification.

In Figure 2 we suggest the structure of the frame for the case when the authentication tag, i.e., $\mathsf{M\text{-}MAC}_{\mathbb{K}^i_{\mathcal{N}}}(\widetilde{m})$, is embedded in the message frame and we also outline the case when it is sent as a separate authentication frame (dashed arrow). In both cases the frame structure consists in the identifier of the message id_{frame} which is the usual CAN ID, the identifier of the source node $\mathcal{N}_i$, a message counter c_{mes}, the message itself m and the authentication tag $\mathsf{M\text{-}MAC}_{\mathbb{K}^i_{\mathcal{N}}}(\widetilde{m})$. Supplementary, in the case of authentication frames, a new counter c_{aut} specifies the number of the authentication frame (intended for protocol variants where there is more than 1 authentication frame for a message). The last bit of the identifier field specifies whether a frame carries an authentication tag or message (1 vs. 0). Separate authentication frames are sent in our experimental setup with CAN capable boards, while in the CAN-FD simulation from CANoe the data frames carries the authentication tag as well (allowing immediate authentication). The size of the message counter c_{mes} could be roughly around 20–40 bits but this greatly depends on the bus speed (which determines the number of frames released each second). For example, in the case of high-speed 1Mbps CAN at most 10–20 thousand messages can be sent each second. Using large counters may lead to an unnecessary waste of resources, which can be avoided if participants have synchronized clocks. In this case, the protocol can operate over fixed time periods, for example of around 1 hour, and a 20 bit counter may be sufficient (a random nonce can be used to uniquely bind messages to each such time frame).

2.3 The main scheme

In previous work [10] we defined the main authentication scheme around a master oriented communication. This was

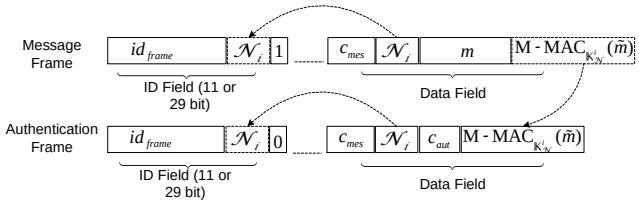

Fig. 2. Data frames and authentication frames

justified by the fact that due to the limited size of a standard CAN frame [19] one frame would not be enough to carry both the message and the authentication tag. Consequently, using a master node with higher computational power to continue the authentication seemed like a correct practical approach, justified also by the results of the experimental section. However, the master oriented communication may somewhat conflict with CAN specifications (which clearly specify that CAN is a multi-master bus) and it also results in more overhead by sending multiple authentication frames (notably, CAN frames have about 50% overhead). Fortunately, as we worked on the protocol, the new CAN standard with flexible data rates CAN-FD was released [20] and this allows us to place all the authentication information in a single frame, reducing the overhead and making it possible to have a cleaner, crisper protocol specification (we also include results from a real-time simulation of the main scheme on CAN-FD).

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. 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* $(\mathsf{Gen}, \mathsf{Tag}, \mathsf{Ver})$ *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* $\mathbb{K} = \{\mathbb{k}_1, ..., \mathbb{k}_s\}$ *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* $\mathbb{M} = (m_1, ..., m_s)$ *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. $v \leftarrow \mathsf{Ver}(\mathbb{k}, m, \tau)$ *is the verification algorithm which takes as input a key* $\mathbb{k} \in \mathbb{K}$*, 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* $\mathbb{k}$*, otherwise the bit* v *is* 0*. For correctness we require that if* $\mathbb{k} \in \mathbb{K}$ *and* $m \in \mathbb{M}$ *then* $1 \leftarrow \mathsf{Ver}(\mathbb{k}, m, \mathsf{M\text{-}MAC}_{\mathbb{K}}(\mathbb{M}))$.

NOTATION. To avoid unnecessary formalism that would not impact security we make some simplifications. Whenever the

TABLE 3
Possible groups with 4 nodes, groups of size 2 outlined

	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}
$\mathcal{N}_1$	0	0	0	0	0	0	0	1	1	1	1	1	1	1
$\mathcal{N}_2$	0	0	0	1	1	1	1	0	0	0	0	1	1	1
$\mathcal{N}_3$	0	1	1	0	0	1	1	0	0	1	1	0	0	1
$\mathcal{N}_4$	1	0	1	0	1	0	1	0	1	0	1	0	1	0

authentication tag does not fit in a single frame we assume that it is sent over separate authentication frames each of them having the proper counter c_{aut} (we do not explicitly use c_{aut} in the description of the schemes since this will only overload the notations). For the same reason, we use the notation $\widetilde{m}$ to denote the message that is already augmented by the counter c_{mes} and node identity $\mathcal{N}_i$ (the node identity $\mathcal{N}_i$ can be eventually skipped if it is embedded in the ID field of the frame). Since, with one exception, all versions of the protocol authenticate the same message to all nodes (rather than authenticate a tuple of messages), we replace $\mathbb{M}$ with the augmented message $\widetilde{m}$ and we write $\mathsf{M\text{-}MAC}_{\mathbb{K}}(\widetilde{m})$. Obviously in this case the M-MAC receives as input a message tuple of s identical messages $\widetilde{m}$, i.e., $\mathbb{M} = \{\underbrace{\widetilde{m}, \widetilde{m}, ..., \widetilde{m}}_{s-times}\}$.

The key allocation procedure distributes the keys to the n nodes by placing them in groups of size g. Figure 4 provides an example of key-sharing for the master oriented version of the scheme, for the main scheme every sender will be placed in the role of the master. We start with an example that shows how we intend to distribute keys to the nodes, then we proceed to the main scheme.

Example 1. Table 3 shows the groups that can be formed in the case of 4 nodes (the empty group G_0 and the group containing all nodes G_{15} are skipped). If we consider groups formed by exactly 2 nodes we have $\binom{4}{2} = 6$ groups (denoted in grey) and each two nodes share exactly $\binom{2}{0} = 1$ group. We also outline the groups shared by $\mathcal{N}_1$, i.e., G_9, G_{10}, G_{12}, and those shared by $\mathcal{N}_2$, i.e., G_5, G_6, G_{12} by marking them with a square. Note that they intersect in one group G_{12}. In Table 4 the case of $n = 8$ is explored, with complete groups of all sizes g and any number of corrupted nodes l. The total number of groups $|G|$ and the number of groups shared by each node $|G'|$ as well as the percentage of uncorrupted keys on each node, i.e., keys that are not known by the adversary, are outlined. This value is computed as follows: for $n = 8$ and $g = 2$ there are 28 groups, if there is only a single corrupted node, i.e., $l = 1$, this is part of exactly 7 groups and controls $7/28 = 25\%$ of the 56 shared keys. But each honest node shares exactly one key with the dishonest one, consequently, out of the 7 keys only 6 cannot be controlled by the dishonest node resulting in a fraction of $6/56 = 21\%$.

Construction 2. *(*LiBrA-CAN - *Main Scheme) Given an* M-MAC *construction for some security parameter ℓ and n nodes placed in groups of size g, we define protocol* $\mathsf{LiBrA\text{-}CAN}_{\mathcal{N}^*}(\mathsf{M\text{-}MAC}, \ell, n, g)$ *as the following set of actions for each CAN node denoted as $\mathcal{N}_i, i = 1..n$:*

1. $\mathsf{Setup}(\ell, n, g)$ *is the key setup procedure. Let $t = \binom{n-1}{g}$ be the number of subsets of g nodes out of the $n-1$ nodes. For each sender, the* Setup *procedure generates t random keys, each of ℓ bits, then distributes to each recevier the keys for the groups that he is part of. For practical reasons, the keys can be distributed in an off-line manner or in an on-line manner by standard techniques, e.g., key-exchange protocols between the corresponding nodes, we do not insist on this since such issues are straight-forward to solve. Let $\mathbb{K}^i_{\mathcal{N}} = \{\mathbb{k}^i_1, \mathbb{k}^i_2, ..., \mathbb{k}^i_t\}$ with $t = \binom{n-1}{g}$ denote the key set of each sender node $\mathcal{N}_i$ and $\mathbb{K}^{i,j}_{\mathcal{N}} = \{\mathbb{k}^i_1, \mathbb{k}^i_2, ..., \mathbb{k}^i_{t'}\}$ with $t' = \binom{n-1}{g-1}$ the key set of each receiver $\mathcal{N}_i$ from the setup procedure of sender $\mathcal{N}_j$.*

2. $\mathsf{SendMesTag}(\mathcal{N}_i, \widetilde{m}, \mathbb{K}^i_{\mathcal{N}})$ *on which node $\mathcal{N}_i$ whenever wants to broadcast a message $\widetilde{m}$ increments its local counter, computes the tag* $\mathsf{M\text{-}MAC}_{\mathbb{K}^i_{\mathcal{N}}}(\widetilde{m})$ *with its keyset $\mathbb{K}^i_{\mathcal{N}}$ and sends the message $\widetilde{m}$ and the authentication tag on the bus.*

3. $\mathsf{RecMesTag}(\mathcal{N}_i, \mathcal{N}_j, \widetilde{m}, \mathsf{M\text{-}MAC}_{\mathbb{K}^i_{\mathcal{N}}}(\widetilde{m}))$ *on which node $\mathcal{N}_i$ receives a data frame containing message $\widetilde{m}$ from node $\mathcal{N}_j$ along with the corresponding tag* $\mathsf{M\text{-}MAC}_{\mathbb{K}^j_{\mathcal{N}}}(\widetilde{m})$. *Node $\mathcal{N}_i$ checks if the message is fresh, i.e., counter up to date, and authentic for all common keys, i.e.,* $1 \leftarrow \mathsf{Ver}(\widetilde{m}, \mathbb{k}, \mathsf{M\text{-}MAC}_{\mathbb{K}^j_{\mathcal{N}}}(\widetilde{m}))$, $\forall \mathbb{k} \in \mathbb{K}^{i,j}$. *If the tag is correct for all keys in the common keyset, i.e., $\mathbb{K}^{i,j}$, then message $\widetilde{m}$ is deemed authentic.*

TABLE 4
Fraction of uncorrupted keys on each node in the case of $n = 8$ participants, groups of size $g = 1..7$ and $l = 0..8$

			Fraction of uncorrupted keys on each node (%)							
g	$\lvert G\rvert$	$\lvert G'\rvert$	l=0	l=1	l=2	l=3	l=4	l=5	l=6	l=7
1	8	1	12.5	12.5	12.5	12.5	12.5	12.5	12.5	12.5
2	28	7	25	21	17	14	10	7	3.5	0
3	56	21	37.5	26	17	10	5	1.7	0	0
4	70	35	50	28	14	5	1.4	0	0	0
5	56	35	62	26	8.9	1.7	0	0	0	0
6	28	21	75	21	3.5	0	0	0	0	0
7	8	7	87	12.5	0	0	0	0	0	0

Example 2. To see that our scheme achieves superior security we compare it to the scheme in [24]. We take for example a setup with 4 receivers and 1 sender. In Szilagy and Koopman's scheme, one frame will be sent containing 4 MACs (one for each receiver). Set each of these tag to 9 bits (in [24] 8 bits are suggested for each MAC, we use 9 here just to make the comparison fair between the schemes, but these numbers are artificially small anyway and serve just as an example). The security level after the first frame is 9 bits for each node and, regardless of the number of subsequent votes from the other nodes, the security level is at most $(4 - l) \times 9$ bits, where l is the number of corrupted nodes. This is because subsequent votes from other nodes will only confirm that their fragment from the initial MAC is correct and there are only $(4 - l)$ trusted fragments each of 9 bits. Now consider LiBrA in the case of groups of size 2 and one sender. In this setup, there are 6 groups and keys shared between the sender and these groups. The sender now has to broadcast 6 MACs each of 6 bits each (this adds up to a 36 bit tag as previously). If there is no corrupted node, as each node has 3 of the 6 keys, the security level after the first message from the sender is double compared to Szilagy and Koopman's scheme, i.e., $50\% \times 36 = 18$ bits vs. 9 bits. Subsequently each new

authentication frame of the sender contributes with the same amount of authentication bits (though our intention is to assure authentication in a single frame in order to avoid delays and reduce overheads which can be done simply by increasing the size of each tag). If we set the number of corrupted nodes to 1, we still get 12 bits of security after a single message and the same amount of authentication bits will be added by each new frame (because for each honest node 33% of the key bits are not corrupted).

The security of the main scheme can be directly linked to the security of the M-MAC construction that we analyse in Appendix A. Below we give an account for the influence of corrupted principals on the security level, i.e., collusion attacks. We simply point to combinatorial bounds as the security of such multicast schemes is well established, see for example [3].

SECURITY. For n participants we have 2^n groups, this includes one empty group and one group which contains all participants. There are $\binom{n}{g}$ possible groups of size g. For n nodes, given all groups of size g, each node is part of $\binom{n-1}{g-1}$ groups. Further, if one considers that there is 1 corrupted node then he shares with each other node $\binom{n-2}{g-2}$ groups. If there are l corrupted nodes then they share with each node $\binom{n-l-1}{g-l-1}$ groups (this gives the number of corrupted keys on each node). For a more accurate view, we translate this discussion into security bits. Assume that the M-MAC is built by simple concatenation of regular MACs (each of them computed with the corresponding shared key). Having an M-MAC of some fixed bit-length t, each individual MAC is truncated to t/n bits. If keys are pair-wisely shared between the sender and receivers, i.e., $g = 1$, then each node receives exactly t/n security bits (since it is in possession of a single key that works for one of the MACs). But in case we share the keys between groups of size g, then each node is in possession of $\binom{n-1}{g-1}$ keys which is a fraction of $\binom{n-1}{g-1} \cdot \binom{n}{g}^{-1}$ from the total number of keys and equivalently from the bits carried by the M-MAC (obviously this more than $1/n$ for $g = 1$).

Figure 3 shows on the left side the decrease in security bits, having fixed 256 bits for the tag, for groups of size $g = 1, 2, 4, 6$ in the case of $0..4$ adversaries (a dishonest minority). If we consider an authentication tag of 256 bits, if this is to be shared by 8 nodes with 8 different keys then only $256/8 = 32$ bits per node remain. Contrary, in the case of groups of size $g = 2$ there are 64 bits for each node; even if there is 1 corrupted node, still 54 bits remain untouched (while this is not much, it may be enough for real-time security). Generally, with 1 adversarial node the highest security is for $g = n/2$ due to the binomial expansion in which the middle coefficient is the larger. With at most $n/2$ adversaries, i.e., adversarial minority, the highest security is at $g = 2$ and it decreases linearly. On the right side, Figure 3 shows the number of groups (which translates into keys) in the case of $n = 1..8$ nodes and subgroups of size $g = 1, 2, 3, 4$. Obviously, this grows exponentially, but for smaller g the number of keys is decent and gives strong security benefits.

We further design several variations of the main authentication scheme that give efficient trade-offs as shown in the experimental results section. These variations can be roughly grouped in two main classes. First, for non-homogeneous networks it makes sense to have a *master oriented authentication* where the node with higher computational power deals with the main part of the authentication procedure. Second, for homogeneous networks where all nodes have similar computational power it makes sense to have a *distributed authentication* scheme where all participants contribute to the authentication task. For brevity, whenever possible without confusions, we skip the formalism related to the schemes.

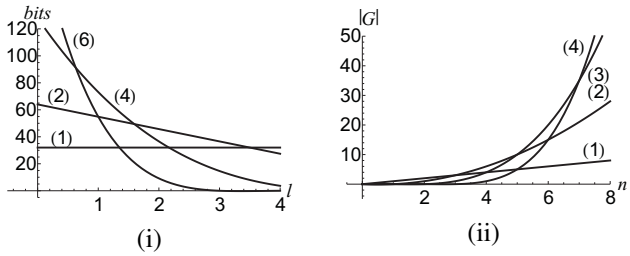

Fig. 3. Fraction of recovered bits (i) for 8 nodes with $l = 1..7$ and number of groups (ii) for $n = 1..8$ (group size given in the round brackets)

2.4 Master oriented versions of the scheme

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. This is also supported by our experimental results. More, if the master node is a trustworthy third party, there are clear security benefits if he handles all keys that are shared between nodes since he can continue the authentication further with all the remaining keys (not only with the keys known to the sender). This is summarized by the next construction. Figure 4 shows the master node and the slave nodes connected to the bus, it also outlines the keys that are shared between nodes.

Construction 3. *(Centralized Authentication) Given an* M-MAC *construction for some security parameter ℓ and n nodes placed in groups of size g, we define protocol* CN-LiBrA-CAN$_{\mathcal{M},\mathcal{N}^*}$(M-MAC$, \ell, n, g$) *as follows. Let $\mathbb{K}_{\mathcal{M}}$ denote the keyset of the master node and run the previous key-setup procedure only for the master node (i.e., the master places the slaves in groups of size g and shares keys with the groups). The following set of actions hold for the master $\mathcal{M}$:*

1. RecMesTag$(\mathcal{M}, \mathcal{N}_i, \widetilde{m},$ M-MAC$_{\mathbb{K}^i_{\mathcal{N}}}(\widetilde{m}))$ *on which master $\mathcal{M}$ receives message $\widetilde{m}$ and authentication the tag* M-MAC$_{\mathbb{K}^i_{\mathcal{N}}}(\widetilde{m})$ *from slave $\mathcal{N}_i$. Subsequently, master $\mathcal{M}$ checks if the counter is up-to-date and if the message is authentic, i.e.,* $1 \leftarrow$ Ver$(\widetilde{m}, \mathbb{k},$ M-MAC$_{\mathbb{K}^i_{\mathcal{N}}}(\widetilde{m}))$, $\forall \mathbb{k} \in \mathbb{K}^i_{\mathcal{N}}$. *If so, he proceeds to authenticating the tag to other nodes with* SendTag$(\mathcal{M}, \widetilde{m}, \mathbb{K}^i_{\mathcal{N}})$.

2. SendTag$(\mathcal{M}, \widetilde{m}, \mathbb{K}^i_{\mathcal{N}})$ *on which master $\mathcal{M}$ gathers all the remaining keys $\mathbb{K}_{\mathcal{M}} \setminus \mathbb{K}^i_{\mathcal{N}}$ computes* M-MAC$_{\mathbb{K}_{\mathcal{M}} \setminus \mathbb{K}^i_{\mathcal{N}}}(\widetilde{m})$ *and broadcasts it as an authentication frame with the same ID as the original message (note that in this case the* M-MAC *is computed with the remaining $\binom{n}{g} - g$ keys, there is no restriction from the construction of* M-MAC *to do it so).*

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

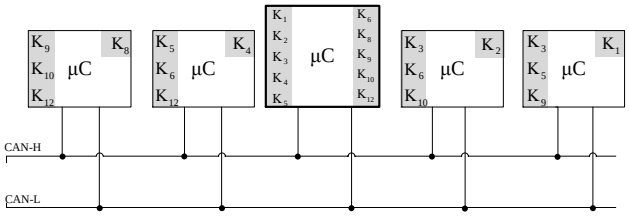

Fig. 4. Master and slave microcontrollers (μC) in a setting for centralized authentication

1. $\mathsf{RecMesTag}(\mathcal{N}_i, \mathcal{N}_j, \widetilde{m}, \mathsf{M\text{-}MAC}_{\mathbb{K}^j_{\mathcal{N}}}(\widetilde{m}))$ *on which slave* $\mathcal{N}_i$ *receives message* $\widetilde{m}$ *and an authentication tag from another slave* $\mathcal{N}_j$ *and proceeds similarly to master* $\mathcal{M}$ *by checking if the message is authentic but only with respect to the keys* $\Bbbk \in \mathbb{K}^i_{\mathcal{N}} \cap \mathbb{K}^j_{\mathcal{N}}$ *that he shares with slave* $\mathcal{N}_j$*. The message is not deemed authentic until a successful* $\mathsf{RecTag}(\mathcal{N}_i, \mathcal{M}, \mathcal{N}_j, \mathsf{M\text{-}MAC}_{\mathbb{K}_{\mathcal{M}} \setminus \mathbb{K}^j_{\mathcal{N}}}(\widetilde{m}))$ *event follows.*
2. $\mathsf{RecTag}(\mathcal{N}_i, \mathcal{M}, \mathcal{N}_j, \mathsf{M\text{-}MAC}_{\mathbb{K}_{\mathcal{M}} \setminus \mathbb{K}^j_{\mathcal{N}}}(\widetilde{m}))$ *on which slave* $\mathcal{N}_i$ *receives an authentication frame containing the tag* $\mathsf{M\text{-}MAC}_{\mathbb{K}_{\mathcal{M}} \setminus \mathbb{K}^j_{\mathcal{N}}}(\widetilde{m})$ *from the master* $\mathcal{M}$ *(that continues the authentication of slave* $\mathcal{N}_j$*) and verifies for all keys* $\Bbbk \in \mathbb{K}^i_{\mathcal{N}} \cap \left(\mathbb{K}_{\mathcal{M}} \setminus \mathbb{K}^j_{\mathcal{N}}\right)$ *if the tag is correct. If for all keys in its keyset the tag is correct then message* $\widetilde{m}$ *is deemed authentic.*
3. $\mathsf{SendMes}(\mathcal{N}_i, m, \mathbb{K}^i_{\mathcal{N}})$ *on which slave* $\mathcal{N}_i$ *whenever wants to broadcast a message* $\widetilde{m}$ *increments its local counter, computes the tag* $\mathsf{M\text{-}MAC}_{\mathbb{K}^i_{\mathcal{N}}}(\widetilde{m})$ *with its keyset* $\mathbb{K}^i_{\mathcal{N}}$ *and sends the data frame containing message* $\widetilde{m}$ *and the corresponding tag on the bus.*

CUMULATIVE AUTHENTICATION. 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. Distinct to the case of the TESLA protocol the delay is not strongly constrained by external parameters (such as oscillator precision, synchronization error, bus speed, etc.).
LOAD BALANCED AUTHENTICATION. The *centralized authentication* scheme is beneficial in the case when the communication master has higher computational resources, but it may be the case that the master node is already busy with other computational tasks. For such case, a *load balanced* version of the scheme can be used in which the communication master can send a flag (authenticated along the message) to point for a particular slave(s) to carry the authentication further.

2.5 Distributed versions of the scheme

Indeed, an authentication master may not always be present. Moreover, several events can lead to his unavailability (for example he can enter Bus Off-mode due to problems with the transceiver or the ECU itself can suffer a malfunction). For this purpose we introduce the cascade authentication scheme where the slaves reply in a cascade manner by sending the authentication tag for their group of keys. In what follows, we assume that all the nodes continue the authentication in a round-robin fashion until they reach the sender (or stop after sufficient authentication tags are released). Thus, we point out to node $(i+1)$ as the next node and whenever we reached the n-th node the first one becomes the next, etc.

Construction 4. *(Cascade Authentication) Given an* M-MAC *construction for some security parameter* ℓ *and* n *nodes placed in groups of size* g*, we define protocol* $\mathsf{DC\text{-}LiBrA\text{-}CAN}_{\mathcal{N}^*}(\mathsf{M\text{-}MAC}, \ell, n, g)$ *as follows. Let* $\mathbb{K}^{i,j}_{\mathcal{N}}$ *denote the keys shared between nodes* i *and* j *(we assume the same key-setup as previously, except that the master does not play any role in authentication), then the following set of actions is defined for each of the nodes* $\mathcal{N}^*$*:*

1. $\mathsf{RecMes}(\mathcal{N}_i, \mathcal{N}_j, \widetilde{m})$ *on which node* $\mathcal{N}_i$ *receives a data frame containing message* $\widetilde{m}$ *from another node* $\mathcal{N}_j$ *checks if the counter is up-to-date then stores the message in a queue of messages to be authenticated.*
2. $\mathsf{RecTag}(\mathcal{N}_i, \mathcal{N}_j, \mathsf{M\text{-}MAC}_{\mathbb{K}^{j,j+1}_{\mathcal{N}}}(\widetilde{m}))$ *on which* $\mathcal{N}_i$ *receives an authentication frame containing tag* $\mathsf{M\text{-}MAC}_{\mathbb{K}^{j,j+1}_{\mathcal{N}}}(\widetilde{m})$ *from another node* $\mathcal{N}_j$ *and verifies for all keys* $\Bbbk \in \mathbb{K}^i_{\mathcal{N}} \cap \mathbb{K}^{j,j+1}_{\mathcal{N}}$ *if the tag is correct. If for all keys in its keyset a correct tag was received then message* $\widetilde{m}$ *is deemed authentic. If* $i = j+1$ *then it proceeds to* $\mathsf{SendTag}(\mathcal{N}_i, \widetilde{m}, \mathbb{K}^i)$*.*
3. $\mathsf{SendTag}(\mathcal{N}_i, \widetilde{m}, \mathbb{K}^i)$ *on which node* $\mathcal{N}_i$ *gathers all the keys shared with node* $\mathcal{N}_{i+1}$ *in the set* $\mathbb{K}^{i,i+1}_{\mathcal{N}}$*, computes* $\mathsf{M\text{-}MAC}_{\mathbb{K}^{i,i+1}_{\mathcal{N}}}(\widetilde{m})$ *and broadcasts it as an authentication frame.*
4. $\mathsf{SendMes}(\mathcal{N}_i, \widetilde{m}, \mathbb{K}_i)$ *on which node* $\mathcal{N}_i$ *whenever wants to broadcast a message* $\widetilde{m}$ *increments its local counter, computes the tag* $\mathsf{M\text{-}MAC}_{\mathbb{K}^{i,i+1}_{\mathcal{N}}}(\widetilde{m})$ *and sends the data frame containing* $\widetilde{m}$ *followed by an authentication frame containing the tag on the bus.*

SECURITY. The cascade authentication is mainly intended for balancing the computational costs of the authentication between principals. Since the nodes proceed in a chain reaction by simply re-authenticating messages with their own keys, new tags are produced that extend the authentication over new keys. The security level at each node is at most that of the tag from the initial message that started the cascade (assuming all nodes in the cascade are trusted, otherwise this is proportional to the number of uncorrupted keys).
TWO-STAGE AUTHENTICATION. In the case of *two-stage authentication* we assume a scenario with nodes of 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 node (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.
MULTI-MASTER AUTHENTICATION. For the same reasons, a distributed version of the *centralized authentication* scheme can be imagined. In this case, several nodes with higher computational power can form a group of communication

masters. Each of them may broadcast a distinct authentication tag and if any such tag is missing, due to the unavailability of a particular node, the other masters will take care of replacing this tag with one of their own.

2.6 Increasing security with LM-MAC

The M-MAC uses 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. We do develop on this by requiring a new property which we call *strong non-malleability* and which lets any verifier detect whenever the adversary had tampered with any part of the M-MAC. We show that both these properties are achievable by the following LM-MAC construction.

Construction 5. *(Linearly Mixed MAC) We define the* LM-MAC *as the tuple of probabilistic polynomial-time algorithms* $(\mathsf{Gen}, \mathsf{Tag}, \mathsf{Ver})$ *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* $\mathbb{K} = \{\mathbb{k}_1, ..., \mathbb{k}_s\}$ *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* $GF(2^b)$*:*

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

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. $v \leftarrow \mathsf{Ver}(\mathbb{k}, m, \tau)$ *is the verification algorithm which returns* 1 *if and only if having* $\tau' = \mathsf{MAC}_{\mathbb{k}}(m)$ *it holds* $\tau' \equiv \mathsf{KD}_1(\mathbb{k}, m) \cdot x_1 + \mathsf{KD}_2(\mathbb{k}, m) \cdot x_2 + ... + \mathsf{KD}_s(\mathbb{k}, m) \cdot x_s$*. 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 [6] 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 for the M-MAC to fail of around 10^{-5} for $b = 16$ and 10^{-10} for $b = 32$.

Example 3. 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 and S_2 along with the authentication tag. To increase the 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, e.g., 64 bits (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 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 feature 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.

For efficiency, we can drop on some of the computation from the M-MAC at small penalties in security. The next construction provides such a simplification.

Construction 6. *(Simplified Linearly Mixed Message Authentication Code) We define the* SLM-MAC *in the same manner as the* LM-MAC *except for the fact that in the generation and verification algorithms the message is not used by the key derivation process, i.e.,* $\mathsf{KD}_i(\mathbb{k}_j, m_j)$ *is replaced by* $\mathsf{KD}_i(\mathbb{k}_j), \forall i, j \in 1..s$*.*

SECURITY. Both these constructions are unforgeable, provided that the underlying MAC is unforgeable. They are also strongly non-malleable and an adversary cannot manipulate a single element of the tag without making the tag fail on all of the underlying keys (except for negligible probability). As the coefficients are constant in the case of the SLM-MAC, this construction will not provide strong non-malleability if the adversary learns any of the $\mathsf{MAC}_{\mathbb{k}_i}(m_j)$ values. This would not be the usual case as the authentication tag is comprised by $\tau = \{x_1, x_2, ..., x_s\}$ from which one cannot build $\mathsf{MAC}_{\mathbb{k}_i}(m_j)$ unless he is in possession of $\mathbb{k}_i$ in order to derive the corresponding coefficients $\mathsf{KD}_j(\mathbb{k}_i), j \in 1..s$. We defer the formal treatment of these properties for Appendix A.

3 Experimental results

To evaluate the performance of the proposed protocol suite, we used several setups with different hardware components having the goal to determine the minimum authentication delay.

3.1 Protocol performance

Automotive grade embedded devices from Freescale and Infineon as well as a notebook equipped with an adapter for CAN communication from Vector were employed. On the

S12X platform, only the main core was used at a frequency of 80MHz (the X stands for the XGATE co-processor that can be further used to optimize computations). The embedded platforms that we used are representative for industry's low-end and high-end edges. We built several test beds as follows:

- *Testbed T1:* $Intel\ T7700 + 4 \times TC1797$. The master node is implemented on a standard PC connected via a high speed CANcab connector from the CANcardXL board to the slave nodes that are built on the TriCore platform (this enables a 1Mbps communication speed).
- *Testbed T2:* $TC1782 + 4 \times TC1797$. Master and slave nodes are built on similar TriCore (TC1797) development boards. CAN communication speed is set to 1Mbps.
- *Testbed T3:* $Intel\ T7700 + 4 \times S12X$. The master node is implemented on a PC (Intel Core2Duo CPU T7700@2.4GHz) while slave nodes are built on the S12X boards. The master-slave CAN communication is done through the CANcardXL using a low speed CANcab for 125kbps.
- *Testbed T4:* $S12X + 4 \times S12X$. Both master and slave nodes are built on identical S12X development boards with CAN communication speed set to 125kbps.
- *Testbed T5:* $S12X + 8 \times S12X$. We used 8 nodes based on S12X boards for cascade and two-stage authentications. To these, one more S12X node was added as master for the case of centralized authentication.

Centralized authentication was tested for the case of 4 nodes in groups of size 2 that leads to a total of 6 groups. Messages and authentication tags are always sent as separate frames and the message size is always 8 bits, e.g., the size of a message from an analog-to-digital converter. The MAC size for each key is truncated 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. While we are aware that MD5 is weak and collisions were found long ago, we use it as a baseline for speed comparisons (since real-time attacks were not reported so far on MD5, it should still be secure for the setting that we address). The resulting hash is then truncated to the desired size. Table 6 holds the timings and bus loads for each test bed. Here δ is the authentication delay, i.e., the delay between the time at which a message is received up to when it is authenticated on the receiver side (this includes the time to verify the tag and partly the time to compute and send it since these two can start as soon as the message is sent). 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 performances 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 $100\mu s$.

A more complex setting was implemented around 8 S12X nodes grouped two by two which leads to 28 groups. The size of the authentication tags and the truncated MAC size differs in each variant. We set up the implementations as follows:

TABLE 5
Example of tag allocation for the cascade scheme DC-8S2F4

-	$\mathcal{N}_1$	$\mathcal{N}_2$	$\mathcal{N}_3$	$\mathcal{N}_4$	$\mathcal{N}_5$	$\mathcal{N}_6$	$\mathcal{N}_7$	$\mathcal{N}_8$
$\mathcal{N}_1$	-	36	10	10	8	-	-	-
$\mathcal{N}_2$	-	-	26	10	10	18	-	-
$\mathcal{N}_3$	-	-	-	18	10	10	26	-
$\mathcal{N}_4$	-	-	-	-	8	10	10	36

- *Centralized:* The sender 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.
- *Cascade:* Three additional nodes take part in the authentication process besides the sender. The sender computes four MACs, three of which are for the nodes that will help to authenticate the message. The helper nodes will then compute three extra authentication tags to provide enough authentication information for all other nodes. An example of tag allocation is suggested in Table 5. On each line 64 bits are distributed to the nodes denoted on the columns (this is achieved by concatenating MACs that are truncated to the corresponding bit length). Here $\mathcal{N}_1$ is the sender and nodes $\mathcal{N}_2$, $\mathcal{N}_3$ and $\mathcal{N}_4$ continue the authentication. If the size of the groups is 2 then each node will get around 36 security bits, but if the size of the groups is increased to 3 then, at the same computational cost and bandwidth, around 63 security bits are received by each node.
- *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 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 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 6 holds the results achieved with these three 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 implementations, a smaller number of MACs are computed some of which are

TABLE 6
Authentication delay and bus-load (various configurations)

Setup and scheme	Tag (bits)	Msg. (bits)	δ (ms)	Bus load	Speed
T1: Centralized n=4, g=2	64	8	0.267	54.31%	1Mbps
T2: Centralized n=4, g=2	64	8	0.378	42.54%	1Mbps
T3: Centralized n=4, g=2	64	8	2.54	53.84%	125Kbps
T4: Centralized n=4, g=2	64	8	1.848	72.22%	125Kbps
T5: Centralized n=8, g=2	64	8	22.62	11.27%	125Kbps
T5: Cascade n=8, g=2	64	8	9.86	36.11%	125Kbps
T5: TwoStage n=8, g=2	64	8	6.806	46.21%	125Kbps

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.

3.2 Computational performance with LM-MAC

Previous results were based on 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.

In Table 7 we give an overview on the computational timings for various hash functions and input sizes on both of the employed platforms. The usual message size for our scenario will be less than 64 bits (the maximum size of the data carried in one CAN frame) since it contains values from various sensors, etc., that are usually small. To this one will need to add the size of the key that is hashed with the message, thus 16 bytes for the input should be the expected length. We also give measurements for a 64 bytes input just to get an upper bound since an input of such size is less likely (this is in fact the maximum size of the CAN-FD data field). For the Linearly Mixed MACs, in addition to the computation of the MACs, two additional 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.

We made a customized implementation dedicated for $GF(2^{16})$ and $GF(2^{32})$ which resulted in a less general but more compact source code without any unnecessary operation (note that more than a single authentication frame is sent, thus security is not limited to a single $GF(2^{32})$ part of a tag). We improved more by noticing that we could perform the same mixing procedure by working in the integer group Z_p where p is Mersenne prime, i.e., a prime of the form $2^q - 1$ for some other prime q, since in this case modular reduction can be performed more efficient. Concretely, for a group size close to $GF(2^{32})$ we chose $p = 2^{31} - 1 = 2147483647$ which allows 31 bits of entropy for any of the mixed tags. Having this fixed, the computational performance was up to about one order of magnitude cheaper than in the case of the $GF(2^{32})$ field. In Figure 5 (i) we make a graphical depiction of the computational costs for the case of $n = 2$, 4, and 8 nodes respectively. In each case the computational costs of mixing the MACs is compared to the computational cost of 2, 4 and 8 MD5 or HMAC-MD5 operations (these are required to compute the authentication tags in any case). The simplified mixing is significantly cheaper compared to the MD5, in the case of Z_p its cost is relatively inexpensive compared to the required hashes. Even in the case of the linear mixing based on Gaussian elimination in Z_p it is still half the cost of the corresponding hashes. Table 8 shows the computational costs of simplified linearly mixing SMIX (based on matrix multiplication and required for the SLM-MAC) in the case of $GF(2^{16})$, $GF(2^{32})$ and $Z_{2^{31}-1}$ and the case of regular mixing MIX (based on Gaussian elimination and required by LM-MAC) in the case of $Z_{2^{31}-1}$. In Figure 5 (i) we make a graphical representations of the costs when compared to the costs of regular MD5s or HMACs required for the corresponding number of nodes. Figure 5 (ii) shows a prediction of the cost increase when comparing the symmetric functions with the additional costs for mixing the tags. The prediction is based on the fact that mixing with linear equations (MIX) is done by Gaussian elimination which is $O(n^3)$ while matrix multiplication by a vector (SMIX) required for simplified mixing is $O(n^2)$. Finally, the number of MD5s or HMACs increases linearly with the number of nodes. Even in the case of $n = 32$ nodes, the cost for mixing the tags is lower than the cost incurred by the corresponding HMACs.

TABLE 7
Execution time for some common hash functions

Function	Input size (bytes)					
	S12X			TriCore		
	0	16	64	0	16	64
MD5	$371\mu s$	$374\mu s$	$689\mu s$	$10.16\mu s$	$11.00\mu s$	$18.34\mu s$
SHA1	$1.144ms$	$1.148ms$	$2.285ms$	$14.64\mu s$	$15.10\mu s$	$27.60\mu s$
SHA256	$2.755ms$	$2.755ms$	$5.440ms$	$41.70\mu s$	$42.35\mu s$	$80.80\mu s$

TABLE 8
Execution time for MAC-mixing (Infineon TriCore)

	SMIX (GF16)	SMIX (GF32)	SMIX (Z_p)	MIX (Z_p)
2x2	$4.205\mu s$	$7.89\mu s$	$2.045\mu s$	$12.50\mu s$
4x4	$15.16\mu s$	$28.80\mu s$	$5.17\mu s$	$36.95\mu s$
8x8	$57.3\mu s$	$111.0\mu s$	$17.28\mu s$	$180.20\mu s$

4 Improvements with CAN-FD and CAN+

There are two main shortcomings with the previous LiBrA-CAN implementation. 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 discuss two methods of eliminating these drawbacks, first by employing the recently released CAN-FD standard and second by using an unofficial extension of the CAN protocol, called CAN+ [28].

4.1 CAN with Flexible Data-Rate (CAN-FD)

Since CAN-FD capable boards are not yet widely available on the market, we take advantage of the CANoe tool from Vector (www.vector.com) which is the industry standard for

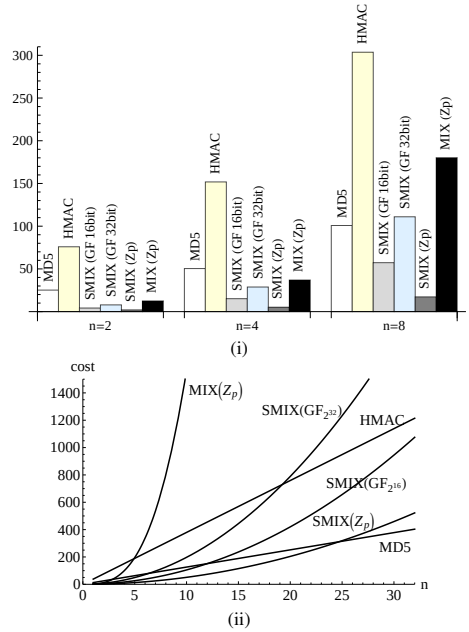

Fig. 5. Computational costs (Infineon TriCore) for mixing the tags (i) and influence of the participants number on computational costs (ii)

developing real-time simulations of in-vehicle networks. We underline that this is a real-time simulation and the software allows connection with real-world networks via various hardware interfaces, e.g., VN1630.

Our simulation considers networks of 4 or 8 nodes in groups of size 2 which send messages in a cyclic manner at 1 or 10 ms. The messages are always 8 bytes long (the maximum allowed in CAN frames) followed by the authentication tag (which shares the same frame with the message in case of CAN-FD). With CAN-FD, the Libra Main Scheme can deliver immediate authentication since the MAC tag can be send with the message in the generous 64 byte data field. Thus there is no authentication delay in our simulation (in contrast to previous experiments from Table 6) since this delay depends solely on the computational power of the node (for which results in Tables 7 and 8 should be taken into account). To emphasize on the benefits of CAN-FD, we compare the bus-load with that of CAN in the case when nodes are broadcasting messages at 1or 10 ms cycles. For the CAN simulation every node sends a message and an additional authentication tag, i.e., 2 frames, at each cycle (immediate authentication is not possible on CAN).

Table 9 illustrates the simulation parameters and the bus-loads obtained for both CAN (running at 1Mbaud) and CAN-FD (running at 1Mbaud with the data segment at 8Mbaud). The first lines show the behaviour for 64 bit tags at various number of nodes, group sizes and cycles, CAN copes with all of these but the bus-load for CAN-FD is at least $4\times$ lower. We also increased the size of the authentication tag to 192 and 448 bits respectively. This allows a very high security level, e.g., in case that no corrupted nodes are present; each node harvests 50% of security bits for $n = 4$ and 12.5% for $n = 8$. CAN-FD behaves perfectly well with bus loads under 25–33%.

TABLE 9
Bus-load at immediate authentication (main scheme)

Scheme	Tag (bits)	Msg. (bits)	Cycle (ms)	Bus load	
				CAN	CAN-FD
Main Scheme n=4, g=2	64	8	1	16.68%	4.41%
Main Scheme n=4, g=2	64	64	10	9.85%	2.07%
Main Scheme n=4, g=2	64	64	1	75.32%	15.98%
Main Scheme n=8, g=2	64	64	10	9.57%	2.03%
Main Scheme n=8, g=2	192	64	1	100%	26.26%
Main Scheme n=4, g=2	192	64	10	19.41%	2.86%
Main Scheme n=8, g=2	448	64	1	100%	33.77%
Main Scheme n=8, g=2	448	64	10	37.97%	4.29%

Note that in the case of standard CAN there is a 100% bus load and some nodes do not manage to transmit messages. In particular, for the case of 192 bits tag only two CAN nodes can send messages along with all the authentication tags, while at 448 bits there is only one node that succeeds so. In contrast to Table 6, since for CAN-FD we don't have separate frames that carry the authentication tag, the bus load covers the entire traffic. Out of this, the authentication tag represents less than $1/2$ for 64 bit tags and is sent in the high data rate segment which works 8 times faster than usual (8 Mbps vs. 1 Mbps). This leads to a bus load of only 33% even in the case of the larger 488 bit tags.

4.2 Backward compatibility with CAN+

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.

The CAN+ protocol [28] allows out-of-band data transmission by inserting extra data into the CAN bitstream in between

CAN sample points. To better understand this, one must first take a step back and take a look at the CAN protocol.

At the data link level, CAN works with a non-destructive arbitration mechanism. The first step of message transmission consists of sending the message's ID. Due to synchronization, all nodes on a bus will start transmission at the same time. If one of the nodes transmits a zero (dominant) bit, the bus will be pulled low, thereby "destroying" the transmission of any one (recessive) bit being transmitted at the same time. Stopping transmission as soon as such a collision is detected leads to the non-destructive arbitration mechanism, which prioritizes low-value IDs. One of the requirements of the arbitration processes is that transceivers detecting a collision do not transmit any more bits, so they need to be able to detect a collision on a bit in the same time window during which that bit is sent. This requires the propagation delay to be less than one clock cycle. However, after the arbitration step, only a single node will still be transmitting and thus, there is no more need to detect bit errors during their transmission window, meaning transmission can happen at an increased rate.

The idea to split CAN message transmission in a slow and fast phase was first proposed by [4]. The CAN+ protocol takes a slightly different approach: CAN messages themselves are transmitted at their usual rate, however, in between sample points, extra data is sent at an increased rate. A traditional CAN transceiver will not notice these extra bits, thereby making the CAN+ protocol function effectively out-of-band. Figure 6 shows the transmission window for CAN+ data inside a CAN bit. Up to 15 extra CAN+ bits can be transmitted along with each CAN bit on a 1 MHz bus. On slower buses, even more bits can be inserted into the data stream. If we assume that CAN+ bits are inserted during the data and CRC field transmission, then a minimum of 225 CAN+ bits can be transmitted (inside the 15 CRC bits, for a 0 byte message on a 1 MHz bus).

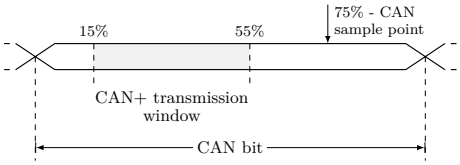

Fig. 6. Timeframe during which the CAN+ protocol can insert extra data into a CAN bit

We implemented a CAN+ transceiver to get a better idea of it's area requirements. The basis of this implementation is revision 1.47 of the open-source CAN transceiver implementation by Igor Mohor, found on the OpenCores website (*http://www.opencores.org*). The design was synthesized with Xilinx ISE 12.2 for a Spartan-6 FPGA with design goal "Area reduction". Our design takes into account the worst case maximum CAN+ data length of 225 bits and, for the given synthesis results, can not transmit or receive more than that. The required area can be found in Table 10. Since it is able to piggyback onto the existing CAN interface logic, the CAN+ transceiver only increases the total transceiver size by a moderate 20%, for a total size of only 430 slices. The total size of the design is smaller than the sum of the size of the two subdesigns (CAN and CAN+), since the Xilinx ISE tool executes compacting operations on the full design. A minimum clock speed of 80 MHz is required for the CAN+ transceiver to function on a 1 MHz CAN bus. Our design meets those requirements with a maximum clock speed of 90.75 MHz.

TABLE 10
Synthesis results for CAN & CAN+ transceiver on a Spartan-6 using Xilinx ISE 12.2 (goal: "Area Reduction").

	Slices	**Registers**	**LUTs**
CAN	499	603	917
CAN+	106	95	227
Combined	430	698	1192

5 Conclusion

LiBrA-CAN is an efficient alternative to achieve source authentication if nodes can be placed in small broadcast groups with dishonest nodes in minority. 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. Experiments performed on the recently released CAN-FD, showed that it is possible to achieve immediate authentication at small costs in bandwidth. If the number of nodes is 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 [9]). CANAuth [26] 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 real-world protocol deployments can be taken only by manufacturers and by means of consortium, we believe that LiBrA carries key concepts for such a deployment. We also consider that the proposal here has the advantage of being simple to implement and simplicity is one relevant criteria for adoption in practice.

Acknowledgment. We thank to Prof. Zonghua Gu for pointing the works of [24] and [25] to our attention. This work was supported by in part by research grants PNII-IDEI 940/2008 and POSDRU 107/1.5/S/77265. It was also supported in part by the Research Council KU Leuven: GOA TENSE (GOA/11/007). In addition, this work is supported in part by the Flemish Government through FWO G.0550.12N and the Hercules Foundation AKUL/11/19, and by the European Commission through the ICT programme under contract FP7-ICT-2011-284833 PUFFIN.

References

[1] H. Bar-El. Intra-vehicle information security framework. In *Proceedings of 9th Embedded Security in Cars Conference, ESCAR*, 2009.

[2] D. Boneh, G. Durfee, and M. Franklin. Lower bounds for multicast message authentication. In *Advances in CryptologyEUROCRYPT 2001*, pages 437–452. Springer, 2001.

[3] R. Canetti, J. Garay, G. Itkis, D. Micciancio, M. Naor, and B. Pinkas. Multicast security: A taxonomy and some efficient constructions. In *INFOCOM'99. Eighteenth Annual Joint Conference of the IEEE Computer and Communications Societies. Proceedings. IEEE*, volume 2, pages 708–716. IEEE, 1999.

[4] G. Cena and A. Valenzano. Overclocking of Controller Area Networks. *Electronics Letters*, 35(22):1923–1925, Oct. 1999.

[5] H. Chan, A. Perrig, and D. Song. Random key predistribution schemes for sensor networks. In *Security and Privacy, 2003. Proceedings. 2003 Symposium on*, pages 197–213. IEEE, 2003.

[6] L. S. Charlap, H. D. Rees, and D. P. Robbins. The asymptotic probability that a random biased matrix is invertible. *Discrete Mathematics*, 82(2):153 – 163, 1990.

[7] 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.

[8] A. Fiat and M. Naor. Broadcast encryption. In *Advances in Cryptology (Crypto93)*, pages 480–491. Springer, 1994.

[9] B. Groza and P.-S. Murvay. Efficient Protocols For Secure Broadcast In Controller Area Networks. *accepted for publication in: Industrial Informatics, IEEE Transactions on*, 2012.

[10] B. Groza, P.-S. Murvay, A. Van Herrewege, and I. Verbauwhede. LiBrA-CAN: a Lightweight Broadcast Authentication protocol for Controller Area Networks. In *Proceedings of The 11th International Conference on Cryptology and Network Security, CANS 2012, Springer-Verlag, LNCS*, 2012.

[11] T. Hoppe and J. Dittman. Sniffing/replay attacks on CAN buses: A simulated attack on the electric window lift classified using an adapted cert taxonomy. In *Proceedings of the 2nd Workshop on Embedded Systems Security (WESS)*, 2007.

[12] T. Hoppe, S. Kiltz, and J. Dittmann. Security threats to automotive CAN networks–practical examples and selected short-term countermeasures. *Computer Safety, Reliability, and Security*, pages 235–248, 2008.

[13] International Organization for Standardization. *ISO 11898-1. Road vehicles - Controller area network (CAN) - Part 1: Controller area network data link layer and medium access control*, 2003.

[14] 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.

[15] J. Leohold. Communication requirements for automotive systems. In *Keynote speech 5th IEEE international workshop on factory communication systems, Vienna, Austria, Vienna University of Technology*, 2004.

[16] M. Naor and B. Pinkas. Threshold traitor tracing. In *Advances in Cryptology (CRYPTO'98)*, pages 502–517. Springer, 1998.

[17] 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.

[18] 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.

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

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

[21] T. Roeder, R. Pass, and F. Schneider. Multi-verifier signatures. *Journal of Cryptology*, 25(2):310–348, 2012.

[22] V. Shoup. Sequences of games: a tool for taming complexity in security proofs. *IACR Cryptology ePrint Archive*, 2004:332, 2004.

[23] 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.

[24] 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.

[25] C. J. Szilagyi. *LOW COST MULTICAST NETWORK AUTHENTICATION FOR EMBEDDED CONTROL SYSTEMS*. PhD thesis, Carnegie Mellon University, 2012.

[26] A. Van Herrewege, D. Singelee, and I. Verbauwhede. CANAuth-a simple, backward compatible broadcast authentication protocol for CAN bus. In *9-th Embedded Security in Cars Conference*, 2011.

[27] M. Wolf, A. Weimerskirch, and C. Paar. Secure in-vehicle communication. *Embedded Security in Cars*, pages 95–109, 2006.

[28] T. Ziermann, S. Wildermann, and J. Teich. CAN+: A new backward-compatible Controller Area Network (CAN) protocol with up to 16x higher data rates. In *DATE*, pages 1088–1093. IEEE, 2009.

APPENDIX A - SECURITY PROOFS

OVERVIEW OF DEFINITIONS AND PROOFS. We now give a formal account of the properties that we require for the M-MAC constructions. These are proved in a framework of two attack games: one for the unforgeability of the M-MAC, i.e., $\mathsf{Game}^{\mathsf{UF}}_{\mathsf{M\text{-}MAC}}$, and the other for the strong non-malleability of the M-MAC, i.e., $\mathsf{Game}^{\mathsf{SNM}}_{\mathsf{M\text{-}MAC}}$. We define these games in similar manner to the games used by Boneh et al. in [2] to derive lower bounds on the security of multicast message authentication. That is, in each game the adversary sends to the challenger a target message and a subset of corrupted parties that collude to fool one or more receivers, further the challenger answers to the adversary's queries and the game ends with the adversary delivering a forged authentication tag. The attack on unforgeability is classical and the proof is straight-forward from the property of the underlying MAC function. The attack on strong non-malleability $\mathsf{Game}^{\mathsf{SNM}}_{\mathsf{M\text{-}MAC}}$ requires an adversary to be able to modify an M-MAC in such way that verification fails with at least one of the keys but succeeds with another. We call an M-MAC that is resilient to such attacks to be *strongly non-malleable*.

Since the unforgeability attack game is easier to derive, we begin with the second property. We prove this by means of a sequence of games following a transition based on the indistinguishability between the key derivation function and some complete random function, a crisper way for providing a security proof [22].

For simplicity of the exposition, we use $\widetilde{m}$ to denote the input to the Tag algorithm and we assume it stands for a tuple of s identical messages, i.e., $\mathbb{M} = \{\underbrace{\widetilde{m}, \widetilde{m}, ..., \widetilde{m}}_{s-times}\}$.

Definition 1. *(Strong Non-malleability Attack Game) We define the* M-MAC *strong non-malleability game* $\mathsf{Game}^{\mathsf{SNM}}_{\mathsf{M\text{-}MAC}}$ *as the following three stage game between challenger $\mathcal{C}$ and adversary Adv:*

1) *Setup stage:*
   a) *Challenger $\mathcal{C}$ runs the key generation algorithm $\mathsf{Gen}(1^\ell, s)$ to get a key set $\mathbb{K} = \{\Bbbk_1, ..., \Bbbk_s\}$ of ℓ-bit keys (ℓ is the security parameter of the scheme).*
   b) *The adversary Adv makes its corruption query by requesting $\mathcal{C}$ a subset of the keyset $\mathbb{K}^{\spadesuit} = \{\Bbbk_{j_1}, ..., \Bbbk_{j_t}\}$, where $j_i \in [1..s]$ and $t < s - 1$. Note that in this case the adversary is always missing at least two of the keys.*
   c) *The adversary Adv fixes its target message $\widetilde{m}^{\blacklozenge}$ and two target key indexes α, β such that $\Bbbk_\alpha, \Bbbk_\beta \notin \mathbb{K}^{\spadesuit}$ (that is, these two keys are not corrupted).*
2) *Query stage:*
   a) *Adversary Adv is allowed to make queries to the MAC generation oracle $\mathcal{O}^{\mathsf{Tag}}(\mathbb{K}, \widetilde{m})$ for any message tuple $\widetilde{m}$ to obtain the corresponding tag $\tau = \{x_1, x_2, ..., x_s\}$ where each x_i is the solution of the following linear system in $GF(2^b)$:*

$$\begin{cases} \mathsf{KD}_1(\mathbb{k}_1, \widetilde{m}) \cdot x_1 + ... + \mathsf{KD}_s(\mathbb{k}_1, \widetilde{m}) \cdot x_s \equiv \mathsf{MAC}_{\mathbb{k}_1}\widetilde{m}) \\ \mathsf{KD}_1(\mathbb{k}_2, \widetilde{m}) \cdot x_1 + ... + \mathsf{KD}_s(\mathbb{k}_2, \widetilde{m}) \cdot x_s \equiv \mathsf{MAC}_{\mathbb{k}_2}(\widetilde{m}) \\ ... \\ \mathsf{KD}_1(\mathbb{k}_s, \widetilde{m}) \cdot x_1 + ... + \mathsf{KD}_s(\mathbb{k}_s, \widetilde{m}) \cdot x_s \equiv \mathsf{MAC}_{\mathbb{k}_s}(\widetilde{m}) \end{cases}$$

b) Adversary Adv is allowed to make queries to the MAC verification oracle $\mathcal{O}^{\mathsf{Ver}}(i, \widetilde{m}, \tau)$ with any key index i, tag $\tau = \{x_1, x_2, ..., x_s\}$ and message $\widetilde{m}$. The oracle $\mathcal{O}^{\mathsf{Ver}}(i, \widetilde{m}, \tau)$ proceeds in the natural way by computing $\tau' \equiv \mathsf{KD}_1(\mathbb{k}_i, \widetilde{m}) \cdot x_1 + \mathsf{KD}_2(\mathbb{k}_i, \widetilde{m}) \cdot x_2 + ...+\mathsf{KD}_s(\mathbb{k}_i, \widetilde{m}) \cdot x_s$ and checking that $\tau' = \mathsf{MAC}_{\mathbb{k}_i}(\widetilde{m})$.

3) Output stage:

a) Eventually, the adversary outputs the pair $(\widetilde{m}^{\blacklozenge}, \tau^{\clubsuit})$ where $\tau^{\clubsuit}$ is a tag maleated by the adversary.

b) The game output is 1 if the following two conditions hold: Ver outputs 1 on $(\alpha, \widetilde{m}^{\blacklozenge}, \tau^{\clubsuit})$ and 0 on $(\beta, \widetilde{m}^{\blacklozenge}, \tau^{\clubsuit})$. Otherwise the game output is 0. Note that in this case, the adversary is allowed to query the MAC generation oracle $\mathcal{O}^{\mathsf{Tag}}(\mathbb{K}, \widetilde{m})$ with $\widetilde{m}^{\blacklozenge}$ to get the correct tag $\tau^{\blacklozenge}$.

Definition 2. *(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 linearly-mixed message authentication code $\mathsf{LM\text{-}MAC}$ is strongly non-malleable.*

Proof. We construct the following sequence of games:

- **Game** $\mathbf{G}_0$. Let this be the original attack game $\mathsf{Game}^{\mathsf{SNM}}_{\mathsf{M\text{-}MAC}}$.
- **Game** $\mathbf{G}_1$. Let this be identical to game $\mathbf{G}_0$ except for the following: in the query stage, whenever solving the linear system of equations replace $\mathsf{KD}_1(\mathbb{k}_\alpha, \widetilde{m}), \mathsf{KD}_1(\mathbb{k}_\alpha, \widetilde{m}), ..., \mathsf{KD}_s(\mathbb{k}_\alpha, \widetilde{m})$ with some pure random values $\rho_1, \rho_2, ..., \rho_s$ (for each $\widetilde{m}$ these are stored on a tape).
- **Game** $\mathbf{G}_2$. Let this be identical to game $\mathbf{G}_1$ except that when verifying the tuple $(\alpha, \widetilde{m}^{\blacklozenge}, \tau^{\clubsuit})$ instead of running $\mathcal{O}^{\mathsf{Ver}}(i, \widetilde{m}, \tau)$ in the usual way the challenger first checks if the adversary ever asked for the correct tag $\tau^{\blacklozenge}$, retrieve it from the tape (if not then run $\mathcal{O}^{\mathsf{Tag}}(\mathbb{K}, \widetilde{m})$ to get it) and checks if: $\rho_1 \cdot (x_1^{\clubsuit} - x_1^{\blacklozenge}) + \rho_2 \cdot (x_2^{\clubsuit} - x_2^{\blacklozenge}) + ... + \rho_s \cdot (x_s^{\clubsuit} - x_s^{\blacklozenge}) \equiv 0$.

Assuming that the probability in distinguishing between the key derivation function and some pure random function is negligible $\epsilon(\ell)$ we have: $\Pr[Adv \text{ wins } \mathbf{G}_0 - Adv \text{ wins } \mathbf{G}_1] < \epsilon(\ell)$.

If $\rho_1 \cdot (x_1^{\clubsuit} - x_1^{\blacklozenge}) + ... + \rho_s \cdot (x_s^{\clubsuit} - x_s^{\blacklozenge}) \equiv 0$ then it also holds that $\rho_1 \cdot x_1^{\clubsuit} + ... + \rho_s x_s^{\clubsuit} \equiv \mathsf{MAC}_{\mathbb{k}_\alpha}(\widetilde{m})$, thus whenever the condition of game $\mathbf{G}_2$ is verified the one for game $\mathbf{G}_1$ is verified as well. It follows that the probability that game $\mathbf{G}_2$ outputs 1 is at least that of game $\mathbf{G}_1$ and we have: $\Pr[Adv \text{ wins } \mathbf{G}_1] \leq \Pr[Adv \text{ wins } \mathbf{G}_2]$. However, note that $\exists i \in [1..s]$ such that $x_i^{\clubsuit} \neq x_i^{\blacklozenge}$ since otherwise $\tau^{\clubsuit} = \tau^{\blacklozenge}$ and in this case $x_i^{\clubsuit}$ will also pass verification for key $\mathbb{k}_\beta$ making the adversary loose the game. Therefore $\exists i \in [1..s]$ such that $x_i^{\clubsuit} - x_i^{\blacklozenge} \neq 0$ and since this is multiplied by some pure random ρ_i value we have:

$$\Pr[Adv \text{ wins } \mathbf{G}_2] = \Pr\left[\sum_{j=1..s} \rho_j \cdot (x_j^{\clubsuit} - x_j^{\blacklozenge}) \equiv 0\right] = \frac{1}{2^b}$$

$$\Rightarrow \Pr[Adv \text{ wins } \mathbf{G}_0] \leq \epsilon(\ell) + \frac{1}{2^b} = \epsilon(\ell) + \frac{1}{2^{p(\ell)}} \leq \mathrm{negl}(\ell).$$

We considered b to be a polynomial $p(\ell)$ in the security level ℓ.

Definition 3. *(Unforgeability Attack Game) We define the $\mathsf{M\text{-}MAC}$ unforgeability game $\mathsf{Game}^{\mathsf{UF}}_{\mathsf{M\text{-}MAC}}$ as the following five stage game between challenger $\mathcal{C}$ and adversary Adv:*

1) Setup stage: is identical to that of the strong non-malleability game $\mathsf{Game}^{\mathsf{SNM}}_{\mathsf{M\text{-}MAC}}$ with the following modification. We request that $t < s$ (that is, the adversary can get all except one of the keys) and the adversary Adv fixes its target message $\widetilde{m}^{\blacklozenge}$ and a single key index α such that $\mathbb{k}_\alpha \notin \mathbb{K}^{\spadesuit}$.

2) Query stage: is identical to that of the strong non-malleability game $\mathsf{Game}^{\mathsf{SNM}}_{\mathsf{M\text{-}MAC}}$.

3) Output stage: the adversary outputs the pair $(\widetilde{m}^{\blacklozenge}, \tau^{\clubsuit})$ where $\tau^{\clubsuit}$ and the game outputs 1 if the following two conditions hold: Ver outputs 1 on $(\alpha, \widetilde{m}^{\blacklozenge}, \tau^{\clubsuit})$ and the adversary never queried $\widetilde{m}^{\blacklozenge}$ to $\mathcal{O}^{\mathsf{Tag}}(\mathbb{K}, \widetilde{m})$. Otherwise the game outputs 0.

Definition 4. *(Unforgeability) We say that a mixed message authentication code $\mathsf{M\text{-}MAC}$ is unforgeable if:*

$$\Pr\left[\mathsf{Game}^{\mathsf{UF}}_{\mathsf{M\text{-}MAC}}(1^{\ell}, s) = 1\right] < \mathrm{negl}(\ell).$$

Theorem 2. *The linearly-mixed message authentication code $\mathsf{LM\text{-}MAC}$ is unforgeable if the underlying MAC is unforgeable.*

Proof. Unforgeability is straight-forward to relate to the unforgeability of a single MAC. In the Setup stage, challenger $\mathcal{C}$ after receiving the index α from the adversary Adv solves all future calls to the generation oracle $\mathcal{O}^{\mathsf{Tag}}(\mathbb{K}, \widetilde{m})$ and verification oracle $\mathcal{O}^{\mathsf{Ver}}(i, \tau, \widetilde{m})$ by using the black box for the MAC. When the adversary outputs $(i, \widetilde{m}^{\blacklozenge}, \tau^{\blacklozenge})$ the challenger yields $\mathsf{MAC}_{\mathbb{k}_\alpha}(\widetilde{m}) = \mathsf{KD}_1(\mathbb{k}_\alpha, \widetilde{m}) \cdot x_1^{\clubsuit} + \mathsf{KD}_2(\mathbb{k}_\alpha, \widetilde{m}) \cdot x_2^{\clubsuit} + ... + \mathsf{KD}_s(\mathbb{k}_\alpha, \widetilde{m}) \cdot x_s^{\clubsuit}$ which is a valid tag if and only if the adversary Adv is successful. Note that similar to the previous games since the adversary is not in possession of $\mathbb{k}_\alpha$ all values $\mathsf{KD}_i(\mathbb{k}_\alpha, \widetilde{m}), i = 1..s$ are simulated by random coins.