Full Paper
MaCAN - Message Authenticated CAN

Oliver Hartkopp, Cornel Reuber
Volkswagen Group Research
{oliver.hartkopp, cornel.reuber}@volkswagen.de

Roland Schilling
Hamburg University of Technology
roland.schilling@tu-harburg.de

Abstract—**The modern vehicle has seen an increase of electronic control systems and the introduction of new interfaces to its networks. Therefore the assumptions on which the security of vehicles have been based in the past have changed and the need for new security concepts on vehicular buses arises. We present an approach for a security extension to the CAN bus which aims at providing message authentication with a minimal bandwidth overhead and full backward compatibility to the existing bus architecture. The security concept uses symmetric authentication measures such as the Cipher-based Message Authentication Code (CMAC) algorithm to authenticate signals and timestamps to guarantee the freshness of the messages. The system is thereby highly flexible and can be adjusted to specific use-cases before and during the runtime. Additionally we describe possible applications for the security functions which can be used to further increase CAN bus security.**

I. Introduction

The number of interfaces of today's cars is continuously growing as we see an increase in Car2X communication research. Additionally new entertainment systems and applications are introduced as part of modern vehicles. In contrast to these new developments the most basic IT infrastructure in a vehicle are still bus systems like the CAN bus, which were first developed in the 1980's. While the reliability and timing advantages of these bus systems are beyond question there are no security systems in place to allow protection of signals (e.g. raw sensor values) on the bus.

In this paper we describe a new security concept which aims at introducing cryptographic authentication procedures on the CAN bus. We thereby concentrate on designing a system which can be easily implemented in a real CAN environment and is fully backward compatible to the existing topology. Therefore we use standard security concepts and adapt them to the specific challenges faced in the vehicular network environment. The proposed concept gives the designer of a vehicular Control Area Network a toolbox at hand to increase the security of the system. The level of security is based on the properties of the application and the available resources. The concept allows to send authenticated signals on the CAN bus which can be verified by either a single node or a group of nodes. It was designed with a focus on low message overhead. As an example a signal and its signature[1] can be sent in one dedicated CAN frame in order to preserve the real-time capabilities of the bus.

[1]We use the expression *signature* to describe symmetric authentication elements in this paper even if it is normally used mostly for asymmetric authentication elements.

In Section II we describe the restrictions that arise when trying to add a security extension to the CAN bus and the requirements we derive from these restrictions for our security concept. Additionally we explain the basic concept and idea behind our security extension and how it meets the requirements mentioned above. Section III gives a detailed explanation of the protocols designed for implementing the security concept on the CAN bus. In the following section we then present applications that may benefit from our security concept and that can be used to increase the security at little extra cost. Section V gives a brief overview of work done so far on the topic of CAN bus security. In the last section we describe possible extensions as well as open questions and present our conclusion.

II. Problem Overview

In this section we identify the main restrictions that apply when trying to add message authentication capabilities to the CAN bus and derive the requirements for the proposed security concept. Based on these we present the basic idea behind our solution and show how it meets these requirements.

A. Restrictions

The following restrictions apply to a security concept on the CAN bus due to its properties as well as the environment in which it is used:

Message length
The standard payload length of a CAN frame is $8\,\mathrm{byte}$. Given a signal length of up to $32\,\mathrm{bit}$ this leads to a maximum signature length of only $32\,\mathrm{bit}$ when signal and signature are being transmitted in one CAN frame. Unlike restrictions such as computational resources that can be increased, the length of a CAN frame is fixed and cannot be extended.
CAN FD, a recently introduced extension to the CAN specification, does allow to transmit up to $64\,\mathrm{byte}$ of payload in one frame [12]. However, it is not clear at this time, whether this protocol will be used to transmit real-time data. We therefore focus our work on the standard CAN specification.

Available resources
The number of available CAN identifiers as well as the

bandwidth of the bus are limited and should be treated as valuable resources. As described by Herrewege et al. [6] the naive use of CAN-IDs in order to run bidirectional protocols on the bus would lead to the requirement of $n \times m$ CAN-IDs, where n is the number of nodes and m the number of messages. As we aim for a backward compatible approach it must be possible to respect the number of available CAN-IDs and the bandwidth restrictions in an existing network. Additionally the CAN-ID and bandwidth consumption should be flexible to allow system designers to adjust it to the resources available in the system.

Bidirectional protocols

Due to the event-driven nature of the CAN bus the use of bidirectional protocols needs to be handled with great care. A physical signal needs to be signed instantly as it occurs. This limits the use of freshness elements to timestamps and counter values. The use of challenges for signal authentication is not advisable as it requires bidirectional communication in time critical situations.

B. Requirements

The following requirements for a CAN bus security concept were identified in order to meet the restrictions outlined above.

Message authentication

Given the increasing number of connections to the internal network of today's vehicles it is crucial to establish methods to guarantee the origin of messages on the bus. Due to the fact that most messages are highly time-dependent, measures against replay attacks are required to protect the freshness of authenticated messages.

Real-time capabilities

Many important signals which need to be authenticated also have real-time properties. Authentication methods should preserve these features. As described by Oguma et al. [11] certain Electronic Control Units (ECUs) are required to respond within $1\ \mathrm{ms}$. To be able to verify the origin of these signals, the signature is required to arrive within this time frame. This cannot be guaranteed if signal and signature are spread across multiple messages.

Flexibility

It should be possible for the designer of a system to apply the security concept to certain signals, while others may need no protection. Additionally the concept should allow authenticating individual signals on-demand without compromising their real-time properties. For signals with weak real-time requirements, multi-frame signatures can be taken into account.

Backward compatibility

It is important that the security concept works together with non security-equipped nodes in order to allow an implementation in already existing vehicular networks. Therefore all signals that were previously available to the network still need to be accessible to all nodes on the bus, even after an authentication element has been added to them.

C. Concept

Based on the small payload size of the CAN frame, asymmetric cryptography is not an option in this application. Therefore the proposed security extension uses Message Authentication Codes (MACs) to establish trust in the origin of signals on the CAN bus. Due to cost restrictions and limited hardware resources the CMAC mode of operation [4] is used, which utilizes a block cipher as the cryptographic element in the MAC. This allows to reuse a hardware block cipher implementation which is also needed for encrypting session keys (SKs).

The message authentication process allows authenticated communication between two ECUs as well as a group of ECUs. Due to the symmetric nature of MACs every member of the group is able to sign and verify messages using the shared group key. Therefore authenticated group communication requires a common trust level inside each group. This trust level may be derived from the origin of the ECU (OEM or third party manufacturer) or the level of hardware security the device offers.

We explicitly do not aim at authenticating every message on the bus, but at presenting a small and flexible concept. In addition to defining permanently authenticated signals it is possible to request signal authentication on-demand. This may be triggered by methods of intrusion detection or plausibility checking on an ECU level.

In the following we legitimate the three main building blocks of our security concept, the level of MAC truncation, the signing protocol and the choice of the freshness element.

1) MAC truncation: To protect the real-time capabilities of the system we chose to truncate the signature down to $32\ \mathrm{bit}$, which can then be sent in addition to the signal in a single CAN frame. MAC truncation is possible due to the low speed of the CAN bus which limits the number of guesses an attacker can perform. To restrict key lifetime, short-lived session keys are used for ECU communication.

The length of the CMAC authentication element is usually identical to the block length of the underlying block cipher. In our test scenario we used the AES block cipher resulting in a CMAC length of $128\ \mathrm{bit}$. In real world implementations more light-weight ciphers, such as *PRESENT* [2] should be used.

The level of truncation which can be applied to the signature is based on the number of guesses an attacker can perform during the lifespan of a session key. NIST defines that the minimum number of bits for a CMAC authentication element should be at least $64\ \mathrm{bit}$ unless special conditions apply [4]. In our case the low speed of the bus together with the use of session keys with a short time to live allow us to reduce the length of the CMAC element to $32\ \mathrm{bit}$. NIST defines that the CMAC length should satisfy the following inequality:

$$T_{len} \geq lg(\frac{MaxInvalids}{Risk}) \qquad (1)$$

where T_{len} is the output length of the CMAC and $MaxInvalids$ is the number of false authentication attempts the attacker can carry out for one key. One access attempt every 20 ms results in 8640000 attempts during a maximum session key lifetime of 48 hours. Therefore the use of a 32 bit CMAC output leads to a risk of 1 in 500 of guessing a single signature during the lifetime of the key. In this case it is still unknown when this forged signature will be accepted. Attempts at such a high frequency can be mitigated by comparing the actual and the expected message frequency on the bus and restricting the number of responses to false signatures.

2) Authentication process: Since the majority of OEM-defined CAN frames contain more than one signal two different formats for message authentication are presented in our security concept. The first is defined as a dedicated CAN frame containing only a signal and its signature as payload. Compared to existing solutions this may increases the number of frames necessary to transmit the same number of signals as the signature occupies additional space. Clearly this option can only be chosen for a limited number of signals since the number of available CAN-IDs is restricted and the message overhead needs to be taken into consideration.
The second format uses a newly introduced CAN-ID and is sent in addition to an unauthenticated standard CAN frame. To simplify the mapping between signal and signature this frame contains both. This format can be used for on-demand authentication of usually unauthenticated signals.

3) Message freshness: To protect the messages against replay attacks the use of challenges is not an option as the event driven nature of many signals prohibits the use of bidirectional protocols. ECUs in the automotive environment can be unavailable for a number of reasons such as sleep cycles. This leads to problematic synchronization issues with regard to counter values as a source of freshness. We therefore decided to use timestamps in order to protect the freshness of messages. To synchronize the time signal between the nodes on the bus a time server (TS) is added to the system which transmits the current timestamp in regular intervals and also allows to request an authenticated timestamp if necessary.
The proposed protocols allow to define which signals are signed and which signature frequency (in relation to the message frequency) is used during the runtime of the system. Therefore the security system can be adapted to the signal (e.g. how fast the signal can change based on its physical properties) and to the resources available in different situations.

III. Protocols

In the following section we describe the three protocols defined for signing physical signals, distributing session keys and synchronizing the time signal on the CAN bus. Additionally the message format used for these protocols is presented.

A. Message format

To allow direct communication between two or more ECUs we define a new identification token for all ECUs which are part of the security system. This 6 bit so called ECU-ID can be used to address and identify individual ECUs. Since the standard CAN frame format does not include any target address fields, we introduce a new partitioning scheme of the CAN frame payload in order to allow direct addressing of ECUs. This format is called *crypt frame* and is depicted in Figure 1. The CAN frame format is not altered. The *crypt frame* is the basis for the messages in the security protocols we define next.
Each node sending a *crypt frame* uses a dedicated CAN-ID defined at design time which is mapped to its ECU-ID using a simple lookup table. This allows to infer the sender's ECU-ID (src_id) from the CAN-ID of a received *crypt frame*.
The *crypt frame* format also includes a destination field which is stored in the six least significant bits of the first payload byte of a standard CAN frame. The remaining two most significant bits of the first payload byte are used as *crypt message flags*. Together with the message length or the address fields these flags are used to distinguish the different message types in the newly introduced protocols.

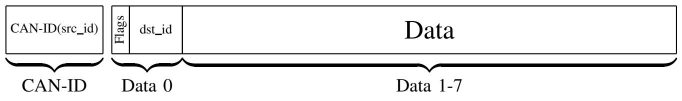

Figure 1. The *crypt frame* format

After the definition of these fields the *crypt frame* format has a remaining payload size of 7 byte. N.B. whenever values or bit fields exceed the length of 8 bit the Little-Endian format is used as payload byte order.
Due to the fact that the src_id defines the CAN-ID of an ECU the *crypt frame* format is fully compatible to the standard CAN frame and allows to use the hardware CAN-ID filters in common CAN controllers to filter for *crypt frames*.

B. Signal Authentication

The standard signal authentication process is outlined in Figure 2.

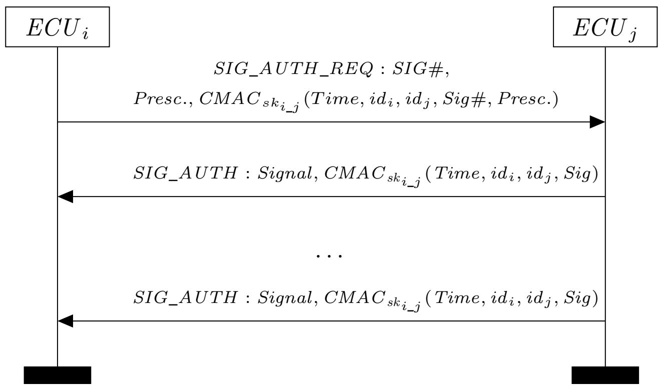
ECU_i
ECU_j
$SIG\_AUTH\_REQ : SIG\#,$
$Presc., CMAC_{sk_{i\_j}}(Time, id_i, id_j, Sig\#, Presc.)$
$SIG\_AUTH : Signal, CMAC_{sk_{i\_j}}(Time, id_i, id_j, Sig)$
...
$SIG\_AUTH : Signal, CMAC_{sk_{i\_j}}(Time, id_i, id_j, Sig)$

Figure 2. Signal Authentication

The purpose of the protocol is to control the authentication process of a signal on the bus. The first message defines the requested signal and the signature frequency in relation to the

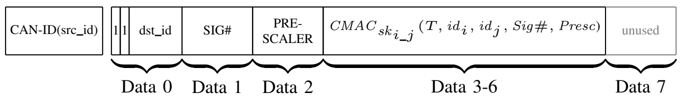

Figure 3. SIG_AUTH_REQ Message

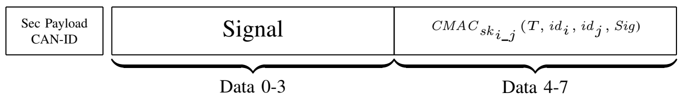

Figure 5. Standard CAN frame with a 32 bit signature

message frequency. This message is depicted in Figure 3. To prevent an attacker from requesting any signed material from another ECU this message is also protected by a signature. The requested signature frequency is defined by the message frequency divided by the value in the prescaler field. In case the prescaler is set to two, a message with a regular interval of 20 ms will be signed every 40 ms. If the prescaler is set to zero the message will only be signed once. For messages without a fixed message frequency (e.g. event driven messages) two different cases need to be considered:

1) If the prescaler is set to zero only the next signal is sent authenticated. All following signals are sent without a signature.
2) For all non-zero prescaler values the signal is signed until the node receives a new control message.

The required signal is defined by the SIG# field. This 8 bit number needs to be defined for every physical signal during the design of the system. This new signal number serves to identify individual signals despite the grouping of multiple signals in a single frame that is common in Control Area Networks.

After the second node receives the control message it sends the signal signed as requested. The signed signal can thereby be sent in the two different formats defined in Section II-C2. The first format is outlined in Figure 4.

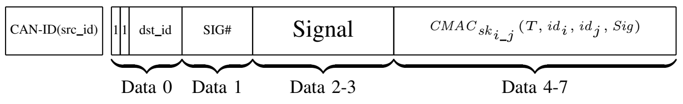

Figure 4. Crypt frame with a 32 bit signature

This message format uses the *crypt frame* layout and should be chosen to transfer an authenticated version of a single signal. Using this message format the signed version of the signal needs to be transmitted in a separate frame in addition to the unauthenticated signal which has been sent on a different CAN-ID before. This does however allow to sign every signal with a size of up to 16 bit.

The second format outlined in Figure 5 is used if a dedicated CAN-ID for authenticated communication can be assigned to a signal number (SIG#) at design time. The SIG# field is omitted in this format and the signal is identified by the CAN-ID (Secure Payload CAN-ID).

This message format should be chosen for signals which need to be signed with a high frequency or which have hard real-time constraints. In this case signals with up to 32 bit can be signed as long as the signature only needs to be verified by a single ECU or a group of ECUs sharing a common group key.

Both signing methods allow to control the signature frequency using the initial request message and then send the signed messages event-driven or periodically.

In case more than one destination requires an authenticated signal an additional *dst_id* field needs to be embedded into the 32 bit *Signal* field in Figure 5. In this case different options to send the authenticated signal are available:

Round robin signing
In case the signature frequency is lower than the message frequency the sender of an authenticated signal may alternately sign the message with different keys.

Increased message frequency
If the signature frequency equals the message frequency the sender can, if technically possible, increase the message frequency and then alternately sign the message with different keys.

Group Keys
As we will see later group keys are available as long as all members of a group share a common trust level. With these group keys it is possible to generate a signature that can be verified by every member of the group. Therefore whenever sender and recipient of the signal are in the same group the sender should use the group session key to sign the message.

C. Key Distribution Protocol

The key distribution protocol is designed to manage session keys across pairs or groups of ECUs. This allows to add and remove ECUs easily from the system and to keep the lifetime of key material short. To provide this service a new instance is added to the bus, the key server (KS) which shares a symmetric long-term key (LTK) with each security-enabled node on the bus. The protocol is depicted in Figure 6.

Since this protocol is carried out only in non time-critical situations challenges are used to assure the freshness of the messages. Before two nodes on the bus are able to perform an authenticated communication with each other, both of them need to request a shared session key from the key server.

To start the key distribution process the first ECU transmits its 6 byte challenge and the 6 bit ECU-ID of the requested communication partner (the so called fwd_id, as is depicted in Figure 7).

The server then generates the session key and sends it in encrypted form to the requesting ECU. Additionally a REQ_CHALLENGE message is sent out by the key server to the second communication partner or to the group as indicated by the *fwd_id* field. Whenever the key server receives a request for a session key from one of the communication partners it

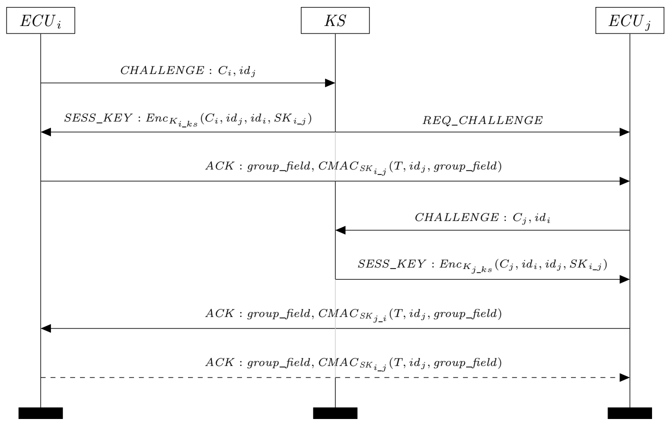
ECU_i
KS
ECU_j
CHALLENGE : C_i, id_j
SESS_KEY : Enc_K_i_ks(C_i, id_j, id_i, SK_i_j)
REQ_CHALLENGE
ACK : group_field, CMAC_SK_i_j(T, id_j, group_field)
CHALLENGE : C_j, id_i
SESS_KEY : Enc_K_j_ks(C_j, id_i, id_j, SK_i_j)
ACK : group_field, CMAC_SK_j_i(T, id_j, group_field)
ACK : group_field, CMAC_SK_i_j(T, id_j, group_field)

Figure 6. Key Distribution Protocol

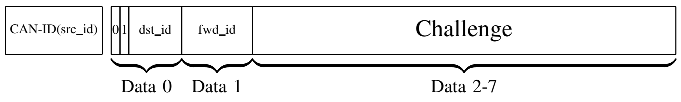

Figure 7. *CHALLENGE* Message

answers with the previously generated session key until the lifetime of this key expires. To overcome the short payload size of the CAN frame we use a simple transport protocol to distribute the session keys. The format of this message is outlined in Figure 8. It is sent exclusively by the key server. Using the AES-Wrap algorithm defined in RFC 3394 [15] a session key message spans six CAN frames.

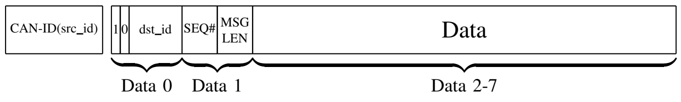

Figure 8. SESS_KEY Message

The session key needs to be requested after every (re)boot of a node or after 48 hours of use. This way no session keys need to be stored beyond the runtime of the system and the use of smaller signatures is possible. Whenever a node received and successfully decrypted a SK it responds with a signed ACK message to the communication partner or group (see Figure 9). This is especially important in group communication since it cannot be guaranteed that all group members receive the key at the same time. In group communication the ACK message includes a 24 bit field indicating the nodes the ECU is aware of sharing this key with. In case another ECU already shares this key but does not find itself in the ACK message it repeats its own ACK message (as indicated by the dashed message in Figure 6). Due to the limited space in the ACK message the maximum number of members in a group is limited to 24. Since communication between a pair of ECUs is a special case of group communication the ACK message is also sent in this scenario. In this case however only the first 2 bit of the group_field are used.

The ACK message shares its *crypt message flags* with the SESS_KEY message. Both messages can be distinguished by their src_id field as SESS_KEY is only sent by the key server.

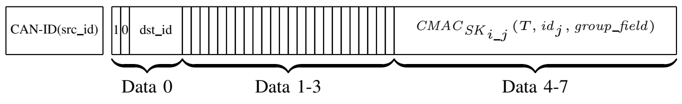

Figure 9. ACK Message

Since symmetric authentication does not guarantee non-repudiation, group authentication requires separate consideration. We therefore propose to partition the set of all nodes in a network into distinct groups sharing a common session key. This way a message signed with the group key can be validated by all nodes in the group. Sharing a common session key, however, all members of a group can assume the role of any other member of the same group. To overcome this problem we propose to form the groups based on the trust level of their members. This minimizes the chance that the signals of a well protected ECU can be forged using the group key it shares with a less trustworthy device. In the key distribution protocol groups are implemented similar to normal nodes. Each group is assigned a 6 bit Group-ID (taken from the ECU-IDs) and the key server as well as all member of the group need to have a list of the group members and their ECU-IDs. Whenever a node requests a session key for communication with another node it first checks if the second node is in the same group. If this is the case it requests the session key not for the ECU-ID of the second node but for the common Group-ID. The key server then sends the challenge request message to all the other group members and all group members can request the common group key.

It is important to mention that all nodes still use their own 6 bit ECU-ID as their src_id and that the Group-ID is only used in the dst_id or fwd_id fields. Since the CAN-ID is used for prioritization as well as for arbitration and the src_id defines the CAN-ID, arbitration errors would be the result if more than one node would use the same src_id. Using the Group-ID as the dst_id and fwd_id the key distribution protocol can be used for groups in the same way it is used for pairs of nodes. Each node that is part of the group needs to filter messages on the bus for messages addressed to their own ECU-ID and to the Group-ID.

D. Authenticated Time Distribution Protocol

The main challenge when using any kind of non bidirectional freshness element on the bus such as a counter or a timestamp is that ECUs may use sleep modes during the runtime of the system or might even reboot due to the harsh runtime and power conditions inside a vehicle. We assume however that every ECU maintains the time signal using its internal counter even if it is not as precise as a real-time clock (RTC) and may need to be resynchronized after the reboot of the ECU. For this purpose a new instance is added to the system, the so called time server (TS). The time server does not need to be a new physical device but may be added to an existing ECU. The main purpose of the time server is to provide a reliable, monotonically increasing time signal to all the nodes on the bus and also offer cryptographic protection of this signal. The message purely consists of a $32\ \mathrm{bit}$ timestamp sent out periodically.

Every node on the bus is able to receive and therefore compare the time signal to its internal clock. If the two signals deviate more than a certain amount or when resynchronization is necessary, e.g. due to a reboot of an ECU, an authenticated time signal can be requested. This can either be the case because the internal clock is not precise enough or because an attacker tries to alter the time signal on the bus. The protocol for this process is depicted in Figure 10.

At first an ECU sends a challenge to the time server in order to request an authenticated time signal. The frame format is the same as in the key distribution protocol. The time server then answers with the last broadcasted timestamp together with a signature over this value and the challenge. In this way also a previously forged message can be detected.

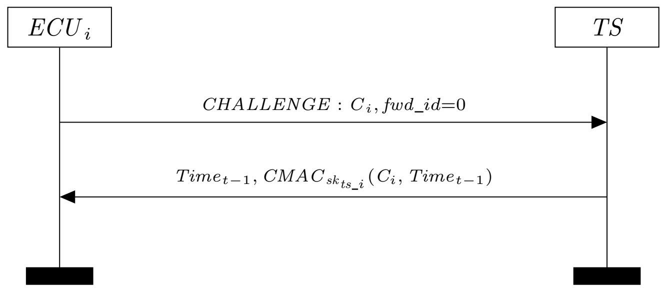
ECU_i
TS
$CHALLENGE : C_i, fwd\_id{=}0$
$Time_{t-1}, CMAC_{sk_{ts\_i}}(C_i, Time_{t-1})$

Figure 10. Authenticated Time Distribution Protocol

The periodically broadcasted time signal and the authenticated time signal are both sent in a standard CAN frame so that every node on the bus can process it. The standard time signal and the authenticated time signal can be distinguished by the data length code of the message.

Since the integrity of the *CHALLENGE* message cannot be verified by all members of a group, replay attacks cannot be prohibited when using group keys in this protocol. Therefore every node has to request an authenticated time signal individually whenever it needs to resynchronize its counter.

Giving only one recipient of the timestamp or a group of recipients with the same key it would not be necessary to use a challenge and response protocol for the purpose of time synchronization. In this case after an initial challenge and response step the signal could be broadcasted together with a signature. Due to the monotonic rising nature of the time signal replay attacks against the time signal authentication process could be prohibited. In our case however this would lead to n different messages for every point in time since each ECU would need its own signature appended to the time signal. Due to these restrictions we decided to use the challenge and response approach described above.

IV. Applications

Besides the obvious application of signing all messages on the bus in order to assert their origin we see other more resource friendly applications for this flexible signing mechanism. One of them is plausibility checking which can be used to extend the trust in cryptographically authenticated messages to other messages without having to sign them.

Here one or more signals can be used to generate a model for another signal in order to compare it with its observed behavior. These techniques are well known from the area of fault detection [5, 13] and can also be used against active attackers as long as measures exist to secure the signals the model relies on. Since all signals are available on the bus anyway the plausibility check does not lead to a significant traffic increase. These techniques may also help to detect physical attacks which take place directly at the sensor (e.g. cooling a temperature sensor) and would normally not be detectable by cryptographic measures. Plausibility checks may help detecting physically altered sensor data as described by Schilling [16].

Additionally plausibility checks based on the physical properties and the dynamic characteristics of a signal can help to gain trust in a signal at limited cost. Thereby trusted points in the development of a signal over time can be used to infer other points in time. A speed signal for instance may only need to be signed at a reduced frequency due to the limited acceleration capabilities of the car.

Other possible applications include the support of intrusion detection systems such as the one proposed by Hoppe et al. [9]. In such a system an authenticated version of a suspicious signal can be requested by an Intrusion Detection System (IDS) instance in order to reassure the integrity of a node.

The proposed security system also allows to sign messages permanently when they are safety relevant signals or selectively when certain conditions apply.

V. Previous Work

Several papers on practical security flaws of in-vehicular bus systems [3, 7, 8, 10, 14] as well as their structural security problems [18] have been published. To protect these systems against such attacks different solutions have been proposed. CANAuth, published by Herrewege et al. [6], is a CAN bus security system which provides message authentication and replay attack resistance. In order to fulfill these requirements symmetric keys and the HMAC algorithm are used. CANAuth also features a key establishment protocol. To authenticate a single CAN frame CANAuth generates up to $15\ \mathrm{byte}$ of security overhead which is sent using the CAN+ protocol [19]. This extension to the standard CAN protocol allows to add additional bits in between the sampling points of a CAN frame. While CANAuth offers various security features it is based on the availability of the extra $15\ \mathrm{byte}$ offered by the CAN+ protocol.

Schweppe et al. [17] propose a security system based on the EVITA hardware security module. The system uses symmetric keys and MACs in order to authenticate messages on the

CAN bus. In combination with the use flags of the Hardware Security Module (HSM) it is possible to use these symmetric keys in an asymmetric fashion. Authentication overheads of 32 to 512 bit are possible in this system. For transmission of these authentication elements the CAN transport protocol ISO 15765-2 is used. Hence for longer authentication elements the real-time properties of the bus may suffer when the message and its authentication element need to be split up in more than one CAN frame.

Bar-El [1] describes a security toolbox which implements several symmetric protocols for authentication and offers a high-level API. This enables a system designer to add security functions to the system without having to deal with their details. The toolbox however does not address any special bus system but rather an abstract vehicular system.

VI. Conclusion

In this paper we described the difficulties that arise when trying to add a security concept to the CAN bus in an automotive environment. As a solution we presented our security concept which allows to authenticate signals on the CAN bus between single nodes as well as within groups. The security concept can be adjusted to existing CAN bus architectures and the new message format is fully backward compatible to the standard CAN frame format. The newly introduced time server helps to guarantee the freshness of signed messages by providing a time signal with an optional signature. The short signature length of 32 bit, which is justified by the low speed of the bus and the restriction of the session key lifetime, allows to retain the real-time capabilities of the bus by sending signal and signature in one CAN frame if necessary.

The security concept is flexible and built with additional security applications such as plausibility checking or intrusion detection measures in mind. In combination with these measures a comprehensive security concept can be built that can help protecting systems with few changes and low overhead using the existent bus infrastructure in today's cars.

In order to securely store the key material all participating nodes need to be equipped with secure memory. Since session keys can be requested at any time from the key server there is no need for the nodes to permanently store any information other than their long-term key.

The implementation of comprehensive plausibility checks and IDSs is an interesting future research topic as it allows adding extra security to the system with limited additional cost and resources. The limitation of our system is clearly the missing ability to sign a message for several other nodes that do not have the same trust level. To solve this problem either asymmetric measures or the use of HSMs which restrict the use of symmetric keys would be necessary. Another interesting future research topic is the implementation of such a system in a real car in combination with a comprehensive formal security analysis of the overall system. Additionally the performance values when used in a real environment could give hints for improvements that would lead to a more efficient use of resources.

References

[1] H. Bar-El. Intra-Vehicle Information Security Framework. In *Embedded Security in Cars 7th*, 2009.

[2] A. Bogdanov, L. R. Knudsen, G. Le, C. Paar, A. Poschmann, M. J. B. Robshaw, Y. Seurin, and C. Vikkelsoe. PRESENT: An ultra-lightweight block cipher. In *Proceedings of CHES 2007*. Springer, 2007.

[3] 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 *Proceedings of the 20th USENIX Security Symposium*, Aug. 2011.

[4] M. Dworkin. NIST special publication 800-38b - recommendation for block cipher modes of operation: The CMAC mode for authentication, 2005.

[5] M. S. H. Versmold. Plausibility Checking of Sensor Signals for Vehicle Dynamics Control Systems. In *8th International Symposium on Advanced Vehicle Control AVEC*, 2006.

[6] A. V. Herrewege, D. Singelée, and I. Verbauwhede. CANAuth - A Simple, Backward Compatible Broadcast Authentication Protocol for CAN bus. In *Embedded Security in Cars 9th*, page 7, Dresden, DE, 2011.

[7] T. Hoppe, S. Kiltz, and J. Dittmann. Security Threats to Automotive CAN Networks — Practical Examples and Selected Short-Term Countermeasures. In *Proceedings of the 27th international conference on Computer Safety, Reliability, and Security*, SAFECOMP '08, pages 235–248, Berlin, Heidelberg, 2008. Springer-Verlag.

[8] T. Hoppe, S. Kiltz, and J. Dittmann. Automotive IT-Security as a Challenge: Basic Attacks from the Black Box Perspective on the Example of Privacy Threats. In B. Buth, G. Rabe, and T. Seyfarth, editors, *Computer Safety, Reliability, and Security*, volume 5775 of *Lecture Notes in Computer Science*, pages 145–158. Springer Berlin / Heidelberg, 2009. ISBN 978-3-642-04467-0.

[9] T. Hoppe, S. Kiltz, and J. Dittmann. Security threats to automotive CAN networks - Practical examples and selected short-term countermeasures. *Rel. Eng. & Sys. Safety*, 96(1):11–25, 2011.

[10] K. Koscher and A. Czeskis. Experimental Security Analysis of a Modern Automobile. In *IEEE Symposium on Security and Privacy*, May 2010.

[11] H. Oguma, A. Yoshioka, M. Nishikawa, R. Shigetomi, A. Otsuka, and H. Imai. New Attestation Based Security Architecture for In-Vehicle Communication. In *GLOBECOM*, pages 1909–1914. IEEE, 2008.

[12] Robert Bosch GmbH. CAN with Flexible Data-Rate, 2011. URL http://www.bosch-semiconductors.de/media/pdf/canliteratur/can_fd.pdf.

[13] O. Rooks. *Softwarebasierte Sicherheitsfunktionen in Drive-by-Wire Fahrzeugrechnern*. Logos Verlag Berlin, 2005. ISBN 9783832508470.

[14] I. Rouf, R. Miller, H. Mustafa, T. Taylor, S. Oh, W. Xu, M. Gruteser, W. Trappe, and I. Seskar. Security and Privacy Vulnerabilities of In-Car Wireless Networks: A Tire Pressure Monitoring System Case Study. In *Proceedings of the 19th USENIX Security Symposium*, Aug. 2010.

[15] J. Schaad and R. Housley. Advanced Encryption Standard (AES) Key Wrap Algorithm. RFC 3394 (Informational), September 2002. URL http://www.ietf.org/rfc/rfc3394.txt.

[16] R. Schilling. Design of a minimal-overhead security concept for controller area networks. Master's thesis, Hamburg University of Technology, 2012.

[17] H. Schweppe, Y. Roudier, B. Weyl, L. Apvrille, and D. Scheuermann. Car2X communication : securing the last meter - A cost-effective approach for ensuring trust in Car2X applications using in-vehicle symmetric cryptography. In *WIVEC 2011, 4th IEEE International Symposium on Wireless Vehicular Communications, 5-6 September 2011, San Francisco, CA, United States*, San Francisco, UNITED STATES, 09 2011.

[18] C. P. M. Wolf. Sicherheit in automobilen Bussystemen. *Automotive-Safety & Security*, 6-7, Oct. 2004.

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