# Plug-and-Secure Communication for CAN

Andreas Mueller and Timo Lothspeich

*Robert Bosch GmbH, Germany*

**Security is a topic of rapidly increasing importance in both automotive as well as industrial applications. This is driven by the current trend towards ubiquitously connected systems, a higher degree of automation, and the increasing openness of systems, with a multitude of interfaces and APIs that an attacker might use for malicious purposes. In today's systems, the communication via CAN is often insecure. Although suitable concepts and cryptographic algorithms are basically available, the distribution of the required (symmetric) cryptographic keys between the involved nodes is still challenging. Currently, the key establishment comes along with either a high logistical / organizational effort or high complexity and/or costs. For that reason, we propose a novel approach for establishing and refreshing symmetric cryptographic keys between different nodes in a CAN network in a plug-and-play manner. Our approach captivates by its simplicity, low complexity and high cost-efficiency, and may be readily implemented without any modifications of standard CAN controllers.**

## I. Introduction

The recent trend towards ubiquitously connected systems – be it cars, factories or buildings – does not only come along with numerous opportunities and benefits, but imposes also serious new security threats with a potentially huge impact. If everything is interconnected with each other and with more and more interfaces and APIs being introduced in order to facilitate innovative services and applications, also the attack surface for malicious manipulations and intrusions is increasing significantly. Without proper countermeasures, hackers may easily take over the remote control of a car, eavesdrop on confidential production data or manipulate a building automation system, for instance.

The fact that this is not just a purely theoretical threat, but rather a real and serious menace, is reflected by various prominent attacks that have been performed and published only recently. In [1], for example, the authors have managed to remotely inject messages on the CAN bus of a Jeep Cherokee (and thus affect important physical systems, such as steering or braking) by exploiting various security flaws and connecting to the vehicle via a mobile network. This led to recall of about 1.4 million cars and fueled legislative initiatives to mandate car manufacturers to support reasonable measures to protect cars against hacking attacks [2]. Further security leaks and successful attacks on cars and other vehicles have been reported in [3] - [5], for example. One of the reasons why especially remote attacks are so relevant and threatening is the fact that these attacks may easily scale and that hackers do not even need physical access to the system under attack. For instance, imagine a scenario with thousands of cars being remotely hijacked and hackers taking control of them. Then, they may precipitate a breakdown of the whole traffic infrastructure of a city or country by manipulating all cars in a coordinated manner. Clearly, this may not only lead to a tremendous physical damage, but also to a significant impact on the whole economy and society. Therefore, the support of appropriate security mechanisms represents without doubt a crucial prerequisite for the success and acceptance of any connected system.

A solid and robust security concept generally covers many different aspects and represents a multi-stage approach, which combines different components. Usually, this includes things like security-aware development processes, fine-grained access control mechanisms and policies, the use of cryptographic methods as well as associated key management procedures. In automotive networks – which we will focus on in the following, even though our approach is readily applicable to many other systems as well – a secured communication on CAN represents a particularly important building block in this respect since a compromised CAN network may have a direct impact on passenger or other people's safety. This is because CAN is typically used for interconnecting all kinds of sensors and actuators, e.g., for powertrain or chassis subsystems. Today, however, the communication on CAN is mostly completely insecure. Even though suitable concepts and algorithms are basically available (such as tailored approaches for authenticating or encrypting CAN messages [6], [7]), they are not used in practice yet as various other challenges still remain open. Among other things, this includes proper standardization across different OEMs and suppliers, but also efficient approaches for establishing, refreshing and managing the cryptographic keys that are required for the involved cryptographic schemes. In this paper, we therefore propose a novel approach addressing the latter aspect, which is able to establish and refresh symmetric cryptographic keys between two nodes in a CAN network in a plug-and-play manner. To this end, special properties of the CAN physical layer are exploited and the approach captivates by its simplicity, low complexity and high cost-efficiency. Moreover, it may be readily implemented with no or only minor extensions to standard CAN controllers. It is particularly suitable to enhance the security against remote (and thus scalable) attacks and thus may become an important building block for secure communication on CAN.

The remainder of this paper is structured as follows: In Section II, we outline our system and attacker model, followed by a review of existing approaches to CAN security in Section III. Our novel approach for establishing and refreshing cryptographic keys based on special CAN properties is then presented in Section IV. In Section V and VI, we discuss certain implementation aspects and elaborate on various security considerations, before concluding the paper with a short summary in Section VII.

## II. System and Attacker Model

In the following, we always consider a setup as depicted in Figure 1. Two devices (Alice and Bob) are connected to the same CAN bus segment and want to establish a pair of symmetric cryptographic keys. Afterwards, they may then use these keys for encrypting and/or authenticating any messages exchanged between them. In addition, however, there may also be a potential attacker (Eve) connected to the same bus segment, which tries to determine or influence the keys to be established between Alice and Bob. In this regard, we make the following assumptions on Alice, Bob and Eve:

1) All nodes have a similar setup, made up of a CAN transceiver, a suitable CAN controller, as well as a microprocessor running the actual application software.
2) Eve is the victim of a remote attack in the sense that the original software running on that node has been replaced by a modified (malicious) software.
3) Eve may eavesdrop on all messages exchanged on the CAN bus. Furthermore, she may inject arbitrary (single) bits on the bus, e.g., by bypassing the CAN controller and directly accessing the CAN transceiver from the malicious software running on the device.

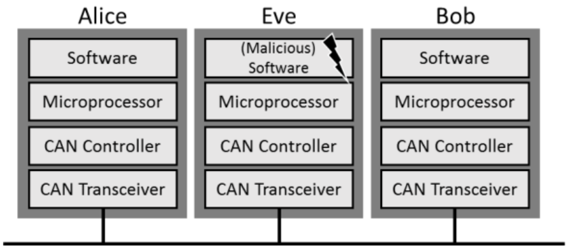

*Figure 1: Considered System Model*

A major challenge in general is to make sure that even if one device has been successfully attacked (here: Eve), the impact on the overall system can be kept to a minimum. In the previously mentioned attack on a Jeep Cherokee, for example, first of all the head unit has been successfully compromised [1]. With proper security mechanisms in place (e.g., proper message authentication), it would not have been possible for the head unit to control safety-relevant functions, such as the brakes of the car, by injecting CAN messages that it actually is not allowed to transmit. However, this is only possible as long as the cryptographic keys of the legitimate nodes (here: Alice and Bob) remain secret. Therefore, Eve naturally must not be able to determine and/or influence these keys.

## III. Review: Security for CAN Networks

The protection of the integrity of a message and the assurance of the authenticity of its sender should generally be among the top security goals in CAN-based networks as CAN is widely used for controlling physical systems or processes with a potential direct impact on safety. Therefore, unauthorized manipulations have to be prevented or they should at least be detectable. Confidentiality, in contrast, is considered to be only of secondary importance and may be useful for making it harder for an attacker to learn the current system state or for delivering critical software updates, for example.

In principle, all these security goals could be achieved in exactly the same way as in the conventional IT world (e.g., using digital signatures, message authentication codes, etc.), but for optimal performance the specific constraints of CAN-based networks should be properly taken into account. This includes things like the limited data rate and message sizes, for example, as well as the limited computational power and memory of many CAN devices. Therefore, in [6] and [7] several security mechanisms specifically optimized for CAN have been proposed, which take these specific constraints into account. In this regard, symmetric cryptographic schemes turn out to be the basis for most of the proposed schemes due to their limited computational complexity and bandwidth requirements. The use of symmetric cryptography, however, requires the availability of symmetric (i.e., identical) keys at the involved nodes and the distribution / establishment of these keys represents a major challenge. Possible options include a manual distribution of keys, e.g., at the end of a production line. This, however, involves a considerable (organizational) complexity and reaches its limitations if one or several devices have already been compromised before being integrated into the network (for example due to an attack performed at the supplier). Besides, an automated refreshment of keys cannot be realized this way. An alternative approach that has been actively discussed and considered in recent years is to use key establishment schemes based on asymmetric cryptography for that purpose, such as the Diffie-Hellman key exchange protocol [8],[9]. Major drawbacks of this approach are the high computational complexity as well as the comparatively large amounts of data that have to be exchanged between two nodes in order to set up a (secure) symmetric key. Besides, it should not be forgotten that the security of the Diffie-Hellman key exchange relies only on the difficulty to efficiently solve the discrete logarithm problem on finite fields or elliptic curves using state-of-the-art methods. Therefore, the approach may become insecure from one day to the other if adequate progress is made in this respect. This could be the advent of a high performance quantum computer, for example.

In the next section, we therefore propose a novel approach for establishing symmetric cryptographic keys between two nodes (or to be more precise: an approach for establishing a shared secret, based on which symmetric keys can be derived), whose security does not rely on hard mathematical problems, but rather on physical properties of the CAN bus. Furthermore, it has an extremely low complexity, low bandwidth requirements and may be readily implemented in practical systems. Finally, it may also be used for efficiently refreshing already established keys, thus making it a very useful and promising building block for future secure CAN networks.

## IV. CAN-Based Key Establishment

The basic idea of our approach is that Alice and Bob agree on a shared secret / key by means of a public discussion using standard CAN messages. In particular, both nodes *simultaneously* transmit appropriate CAN frames, so that Eve is only able to see the superposition of both messages, without knowing the exact content of each of them. However, since Alice and Bob themselves know what they have transmitted and since they can see the superposition of both messages as well, they may readily conclude what the respective other peer has transmitted and thus establish a shared secret that is not known to Eve.

For the concrete realization, we rely on the characteristic property of the CAN bus that bit ‘0’ is dominant and bit ‘1’ is recessive, which represents also the basis for the classical bus arbitration. In fact, if Alice and Bob simultaneously transmit a certain bit (as required for our approach), there are in total four different cases that may occur. These are put together in Table 1. Clearly, if one of the two nodes transmits a dominant bit (‘0’), also the effective bit on the CAN bus is a ‘0’ and only if both nodes transmit a recessive bit (‘1’), we also have the recessive state after the superposition on the CAN bus. Therefore, the CAN bus may be considered as a logical AND function of the individually transmitted bits.

*Table 1: Possible Combinations of Dominant and Recessive Bit Transmissions.*

| Alice | Bob | Effective Bit on CAN Bus |
|---|---|---|
| 0 | 0 | 0 |
| 0 | 1 | 0 |
| 1 | 0 | 0 |
| 1 | 1 | 1 |

The actual procedure for agreeing on a shared secret between Alice and Bob is a multi-step approach as follows:

1) Alice and Bob generate independently of each other random bit strings $R_{Alice}$ and $R_{Bob}$ of a pre-defined length $N$.

Example (for $N = 10$):
$R_{Alice}$ = 0 1 1 0 1 0 0 1 0 1
$R_{Bob}$ = 1 0 1 1 0 1 0 1 1 0

2) Alice and Bob extend these random bit sequences in such a way that after each bit the corresponding inverse bit is inserted, leading to the modified bit sequences $S_{Alice}$ and $S_{Bob}$ of length $2N$.

Example:
$S_{Alice}$ = 01 10 10 01 10 01 01 10 01 10
$S_{Bob}$ = 10 01 10 10 01 10 01 10 10 01

3) Alice and Bob simultaneously transmit the bit sequences $S_{Alice}$ and $S_{Bob}$, leading to the superimposed (effective) bit sequence $S_{eff}$ on the CAN bus, which is given as $S_{eff} = S_{Alice}$ AND $S_{Bob}$.

Example:
$S_{eff}$ = 00 00 10 00 00 00 01 10 00 00

Clearly, Eve may easily determine this bit sequence as well by means of simple passive eavesdropping on the channel. Therefore, it does not really help us further, yet.

4) Alice and Bob determine all tuples in $S_{eff}$ which include a ‘1’.

Example: In $S_{eff}$ given above, there is a ‘1’ in tuples number 3, 7 and 8 (assuming that we start counting the tuples with 1).

5) Alice and Bob delete the bits in their original random bit sequences $R_{Alice}$ and $R_{Bob}$ corresponding to the tuples which included a ‘1’ as determined in step 4. The result are two shortened bit sequences, denoted as $K_{Alice}$ and $K_{Bob}$.

Example: Based on the outcome of step 4, Alice and Bob have to delete the bits at positions 3, 7 and 8 in their original bit sequences $R_{Alice}$ and $R_{Bob}$. Hence, we get:

$K_{Alice}$ = 0 1 ~~1~~ 0 1 0 ~~0~~ ~~1~~ 0 1 = 0 1 0 1 0 0 1

$K_{Bob}$ = 1 0 ~~1~~ 1 0 1 ~~0~~ ~~1~~ 1 0 = 1 0 1 0 1 1 0

Please note that this is done because whenever the effective bit on the CAN bus is a ‘1’, it is clear that both Alice and Bob must have transmitted a ‘1’. Likewise, since the two bits in a tuple are always inverse to each other (cf. step 2), it is also clear that both nodes must have transmitted a ‘0’ for the other bit in that case. However, exactly the same conclusion can also be drawn by Eve and therefore the tuples including a ‘1’ do not provide any usable information for us as no secrecy is contained. For that reason, these bits are simply removed from $S_{eff}$.

6) The resulting (shortened) bit sequence of Alice ($K_{Alice}$) is now exactly the inverse of the corresponding bit sequence of Bob ($K_{Bob}$), which eventually is the established shared secret.

Clearly, what remains after step 5 are the bits that are different in the initial bit strings $R_{Alice}$ and $R_{Bob}$. When simultaneously transmitting $S_{Alice}$ and $S_{Bob}$, we always get ‘00’ for the tuples corresponding to these bits. Hence, by eavesdropping on $S_{eff}$, Eve only knows that Alice and Bob have inverse bits in their original random bit sequences $R_{Alice}$ and $R_{Bob}$ at that position, but she is not able to tell which one has the zero and which one has the one. Alice and Bob, in contrast, know which bit they have transmitted themselves, they can also conclude that the respective other peer has transmitted the inverse bit by evaluating $S_{eff}$ and therefore they have a clear advantage compared to Eve. Thus, $K_{Alice}$ and $K_{Bob}$ are unknown to Eve, but are known by Alice and Bob.

*Discussion*

With the proposed scheme it is possible to establish a shared secret between Alice and Bob by means of a simple public discussion, i.e., by simply transmitting and receiving CAN frames and interpreting the superimposed frames on the bus in the right way. Consequently, the involved complexity is extremely low, especially compared to existing key establishment schemes, such as the Diffie-Hellman key exchange protocol. Yet, it may be done in a fully automated manner and is thus clearly superior compared to the manual distribution of keys.

The concrete integration of the core idea in a full-blown solution with suitable protocol mechanisms is still ongoing work and not elucidated in more detail here due to space constraints. In particular, for a complete solution additional mechanisms are required, e.g., for triggering the synchronized transmission of Alice and Bob, for initiating the whole procedure, and for somehow addressing the involved nodes, for example. In general, however, we do not expect any showstoppers in this respect and for most aspects solid ideas are already available.

In a practical realization, the simultaneous transmission of the bit strings $S_{Alice}$ and $S_{Bob}$ preferably would be done in the payload part of a CAN frame, thus representing a deviation from standard CAN, where simultaneous transmissions may only occur during the arbitration phase when transmitting the CAN identifiers. With some other smart ideas, however, it is still possible to implement the proposed in such a way that other noninvolved nodes (apart from Alice and Bob) see always valid CAN frames on the bus (even with superimposed random bit strings in the payload field) and therefore would not trigger the transmission of any error frame. This will be presented in more detail in the next section. What is important to note, though, is that the number of payload bits in a CAN frame is limited to 64 bits in case of standard CAN and 512 bits in case of CAN FD. Furthermore, the length of the effective shared secret that we can generate with one run of the proposed procedure for a given length *N* of the initial random bit strings $R_{Alice}$ and $R_{Bob}$ is not constant, but depends on how many values in $R_{Alice}$ and $R_{Bob}$ are equal. Clearly, this may vary between zero and *N*, with an expected value of *N/2*. Since in step 2 the initial random sequences are extended by a factor of two by inserting always the inverse bit after each bit and since all these *2N* bits have to be transmitted over the CAN bus, the overall efficiency ρ, which relates the length of the usable shared secret after one round of the proposed approach (given by the length of $K_{Alice}$ and $K_{Bob}$, respectively) to the number of required bits to establish this shared secret (given by *2N*) is generally given by

$$0 \leq \rho \leq \tfrac{1}{2}, \qquad (1)$$

with an expected value of E[ρ] = ¼. This means that on average four payload bits have to be simultaneously transmitted by Alice and Bob in order to establish one secret bit. Since for achieving state-of-the-art security usually symmetric keys of length 128 bit or even 256 bit are required, it is quite clear that for both standard CAN and CAN FD a single run of the proposed approach is generally not enough to generate a sufficient number of secret bits. Therefore, also for addressing this issue suitable protocol mechanisms are required, which ideally would enable the generation of keys of arbitrary length. This may be done by repeatedly performing the proposed procedure and combining the secret bits generated during each run in an appropriate way.

Another very promising application of the our approach is to use it not just for generating full keys of 128 of 256 bits length, but for periodically refreshing existing keys. This is generally beneficial in order to limit the time during which a certain key is used or equivalently the number of messages that are secured using one particular key. By doing so, certain attacks become more difficult (e.g., plain-text attacks) and the potential damage in case that a particular key is revealed at some point in time can be limited. Therefore, periodic key refreshment is a highly recommended security practice in general, see for example [10] and [11]. For refreshing a key, however, already a limited number of secret bits is sufficient as they may be combined with the old key in an appropriate way. This may be done by using a cryptographic hash function, for example. Hence, the proposed procedure may be regularly inserted into the regular CAN communication in order to generate new shared secret bits and to refresh the used keys accordingly for increasing the security level. The periodicity of the key refreshment may be adaptively adjusted depending on the respective needs, thus making it a very flexible and powerful solution in practice.

Finally, it should be noted that CAN is a multicast-based communication protocol and messages transmitted by one node generally have to be received by multiple nodes. This implies that in many cases not only the communication between two nodes has to be secured, but rather the communication between groups of several nodes. Hence, cryptographic schemes for message authentication, encryption, etc. may only be reasonably applied in these cases if all devices belonging to a certain group are in possession of the same cryptographic key. The procedure proposed in this paper, however, cannot be extended to a multi-node setup in a straightforward manner. Nevertheless, there are still several possibilities how so-called *group keys* may be established. In the simplest case, all nodes of a certain communication group could establish a pairwise key with one particular node of that group (e.g., a gateway node), and then this node may generate a suitable group key and signal it to all nodes of the group in a secure manner using the previously established pairwise keys.

## V. Implementation Aspects

As already outlined in the previous section, the bit strings $S_{Alice}$ and $S_{Bob}$ are preferably transmitted in the payload field of a CAN frame. Without any additional measures, however, this may lead to problems and/or compatibility issues in practical realization. In particular, with a direct implementation of the proposed approach, the superimposed CAN frame on the bus may violate the bit stuffing rule since even if the individual bit strings $S_{Alice}$ and $S_{Bob}$ adhere to this rule, it cannot be assured that this is also the case for the effective bit string $S_{eff}$ on the bus. For example, if $S_{Alice}$ = 01010101 and $S_{Bob}$ = 10101010, both strings would be valid, but $S_{eff}$ = $S_{Alice}$ AND $S_{Bob}$ = 00000000 would clearly violate the bit stuffing rule. Hence, other nodes may generate an error frame if they observe such a violation on the bus and clock resynchronization may become more difficult. A relatively simple solution to fix this problem is to insert a fixed bit change (‘01’ or ‘10’) in both $S_{Alice}$ and $S_{Bob}$ after each sequence of at most four bits. This way, Alice and Bob would always transmit the same two bits at this position and since the two bits include a bit change, the bit stuffing rule is never violated in the error-free case. However, this would come at the cost of a higher overhead, of course. Alternatively, Alice and Bob could determine on-the-fly when it is necessary to insert a stuff bit. In fact, both nodes have to read back the effective bit sequence $S_{eff}$ anyway and thus they could check in real-time if there have been five identical bits effectively on the bus and dynamically insert the inverse bit afterwards in that case. Compared to the first solution, the additional overhead would be significantly lower, but in return the complexity and processing requirements are somewhat higher.

A similar problem occurs with the cyclic redundancy check (CRC) field of a CAN frame in case of a direct implementation of the proposed approach. Since $S_{eff}$ depends on both $S_{Alice}$ and $S_{Bob}$, the valid value for the CRC field of the effective message on the bus would generally not be equal to the superimposed CRC fields of the messages transmitted by Alice and Bob in case that they calculate the CRC field in such a way that the transmitted frames are valid. In order to solve this issue and thus assure full backwards compatibility to standard CAN, the correct CRC value that matches to the effective CAN frame on the bus could also be calculated by both nodes on the fly and then be appended to that frame after the payload field. While making sure that the effective frame on the bus is a valid CAN frame (based on existing specifications), it has the nice side effect that this procedure would automatically help to make sure that both Alice and Bob have received the same effective bit string $S_{eff}$ (which is essential for deriving the same shared secret). This is because if one of the two nodes has received at least one erroneous bit but the other one hasn’t, they would append different CRC fields in general, which may be detected by one of the nodes if a recessive bit is overwritten by a dominant one.

With the previously described approaches for dealing with bit stuffing violations and the CRC field, it is possible to achieve full backward compatibility in the sense that all frames on the CAN bus are always in line with the existing specifications – at least in the error-free case. Hence, smooth migration paths become possible, where not all nodes connected to a CAN bus necessarily have to support the proposed approach and existing hardware / software may widely be reused. A practical implementation may be done solely in software (using CAN/GPIO re-pinning, for example, and thus directly accessing the CAN transceiver from the microprocessor) or hardware-assisted, where an additional hardware module may take care of the specific requirements of the proposed approach, such as the synchronized transmission of Alice and Bob. In this regard, we envision flexible implementation options, where existing CAN controllers taking care of the regular CAN communication do not have to be modified at all as long as they are supplemented by an additional (lightweight) hardware / software module.

## VI. Security Considerations

If Eve as modeled in Section II is only passively eavesdropping on the CAN bus, she is not able to readily determine the established shared secret bit sequences $K_{Alice}$ or $K_{Bob}$. As already outlined in Section IV, she just knows that the remaining bits were different for both nodes, but – unlike Alice and Bob – cannot tell who has transmitted the zero and who the one. If, in contrast, Eve is trying to perform an active attack, for example by sending additional own bits during the exchange of $S_{Alice}$ and $S_{Bob}$ between Alice and Bob, there are two different possibilities that have to be considered:

1) Eve is transmitting a recessive bit
2) Eve is transmitting a dominant bit

Transmitting a recessive bit is no different from not transmitting at all since a recessive bit does not change the effective state on the CAN bus. Therefore, we only have to analyze what Eve might do by superimposing another dominant bit to the bits exchanged between Alice and Bob. To this end, it is important to remember that with the procedure proposed in Section IV only those bits remain in the final shared secret for which the effective bit on the CAN bus was '0' for both the transmission of the original and the inverse bit (cf. step 5). Moreover, if Eve transmits a dominant bit, she cannot tell what the status on the CAN bus would have been without her transmission. Therefore, we may conclude the following:

<u>Conclusion 1</u>: An active Eve may disturb our procedure in such a way that the generated secret bit strings $K_{Alice}$ and $K_{Bob}$ are actually not equal on both sides. In order to be able to detect such cases, therefore additional mechanisms should be introduced, with which Alice and Bob can verify that they have really generated the same secret bit sequences $K_{Alice}$ and $K_{Bob}$. This could be done by calculating and exchanging a hash value of these bit sequences, for example.

<u>Conclusion 2</u>: An active Eve is not able to enforce the generation of a particular shared secret between Alice and Bob (which she then would be aware of) and/or to learn the shared secret that is established between both nodes. This is because $K_{Alice}$ and $K_{Bob}$ depend not only on $S_{eff}$ (which may be determined and influenced by Eve), but also on the bits of $R_{Alice}$ and $R_{Bob}$, which are unequal on both sides, and Eve has no way to determine these bits.

<u>Conclusion 3</u>: An active Eve may easily perform a denial-of-service attack by completely preventing the establishment of a shared secret, for example by continuously sending a dominant bit. However, this threat exists basically for any scheme since an active Eve may easily block any CAN communication on the bus. In this case, the fail-safe mode of all devices should prevent any serious safety-critical impact.

If we deviate from the attacker model introduced in Section II and consider not only remote attacks, but also attacks with direct access to the CAN bus (e.g., using own high-end equipment), the situation is getting more challenging. Without any further ado, a passive eavesdropper might be able to determine the shared secret established between Alice and Bob in this case by analyzing the voltage levels on the CAN bus, for example. This is because for fixed positions of Alice, Bob, and Eve, the voltage level that Eve can observe on the bus may be different depending on whether Alice is transmitting a dominant bit and Bob a recessive one or vice versa. For the remote attacker case, this was not an issue since she may only access the CAN bus via a standard CAN transceiver, which regenerates the voltage levels. A remedy could be to artificially introduce a random jitter in the transmit voltage levels of Alice and Bob (within the allowed ranges), so that Eve can no longer conclude who has transmitted which bit. It should also be noted, however, that with direct (physical) access to the CAN bus, an attacker might manipulate a car with much less effort, e.g., by simply cutting through a cable or manipulating the brakes. Nevertheless, a more detailed analysis of potential attacks with direct physical access to the CAN bus as well as possible countermeasures is part of our future work.

## VII. Conclusion and Way Forward

Security will play a crucial role for the success and widespread acceptance of connected systems, such as connected cars and other vehicles. A major challenge in this regard is how to distribute and manage the cryptographic keys between the involved nodes. We have proposed a novel approach for establishing and/or refreshing symmetric cryptographic keys between two CAN devices in a plug-and-play manner, exploiting special properties of the CAN bus. The proposed scheme requires only the simultaneous exchange of random bit sequences along with an appropriate interpretation of the resulting effective bit sequence on the bus. Therefore, it is of very low complexity and may be readily implemented and integrated in practical systems. Even though it certainly cannot address all existing security challenges, it has the potential to become a major building block for secure CAN communication in future. Also, it should be noted that exactly the same concept may also be used in conjunction with other bus systems having similar properties as CAN. Apart from all CAN derivatives, such as TTCAN or CAN FD, this includes the LIN- and I2C-bus, for example.

As a next step, the basic idea has to be embedded in a larger framework, including suitable protocols and mechanisms for synchronized frame transmissions between Alice and Bob, the establishment of group keys, the generation of keys of arbitrary lengths and the like. Furthermore, a first practical proof-of-concept demonstration is planned. In general, however, no major showstoppers are expected in this respect.

**Dr.-Ing. Andreas Mueller**
Robert Bosch GmbH
Corporate Sector Research and Advance Engineering (CR/AEH4)
Robert-Bosch-Campus 1
71272 Renningen, Germany
Phone: +49-711-811-20836
Email: andreas.mueller21@de.bosch.com
Web: www.bosch.com

**Timo Lothspeich**
Robert Bosch GmbH
Automotive Electronics (AE-BE/EKE)
Mittlerer Pfad 9
70499 Stuttgart, Germany
Phone: +49-711-811-34016
Email: timo.lothspeich @de.bosch.com

## References

[1] C. Miller and C. Valasek, “Remote exploitation of an unaltered passenger vehicle”, in *Proc. Black Hat USA*, Aug. 2015.

[2] T. Fox-Brewster, “SPY car act hopes to save American cars from digital disaster”, Online: http://www.forbes.com/sites/thomasbrewster/2015/07/21/senators-launch-spy-car-act/, Forbes, Jul. 2015.

[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 *Proc. USENIX*, Aug. 2011.

[4] 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 *Proc. IEEE Symp. Security and Privacy*, May 2010.

[5] J. Petit and S. E. Shladover, “Potential cyberattacks on automated vehicles”, in *IEEE Trans. Intelligent Transportation Systems*, vol. 16, no. 2, pp. 546 – 556, April 2015.

[6] C.-W. Lin and A. Sangiovanni-Vincentelli, “Cyber-Security for the Controller Area Network (CAN) Communication Protocol”, in *Proc. 2012 Int. Conference on Cyber Security*, June 2012.

[7] B. Glas, J. Guajardo, H. Hacioglu, M. Ihle, K. Wehefritz and A. Yavuz, “Signal-based automotive communication security and its interplay with safety requirements”, in *Proc. escar Europe*, Nov. 2012.

[8] W. Diffie and M.E. Hellman, “New directions in cryptography”, in *IEEE Transactions on Information Theory*, vol. 22, pp. 644-654, Nov. 1976.

[9] A.J. Menezes, P.C. van Oorschot, and S.A. Vanstone, “Handbook of applied cryptography”, CRC Press, October 1996.

[10] H. Krawczyk, M. Bellare, and R. Canetti, “HMAC: keyed-hashing for message authentication”, *RFC 2104*, IETF, Feb. 1997.

[11] S. Bellovin and R. Housley, “Guidelines for cryptographic key management”, *RFC 4107*, IETF, Jun. 2005.