CryptoCAN – Ensuring Confidentiality in Controller Area Networks for Agriculture

Till Zimmermann[1], Jan Bauer[2], Nils Aschenbruck[3]

Abstract: The Controller Area Network (CAN) bus is widely used in existing machinery. Facing more and more vertical integration with more complex devices and integration into public communication networks, its nature as a broadcast-only system without security measures poses serious risks to confidentiality of transmitted data. In this paper, we propose a *Lightweight, Length Preserving and Robust Confidentiality Solution (LLPR-CS)* to retrofit encryption in existing systems, while maintaining full interoperability with these systems. The overhead of our approach is negligible. Therefore, it can be used with existing hardware. By reinterpreting unused bits in the CAN frame format of the ISO 11898 standard, it is possible to build a fully transparent encrypted tunnel in non-confidential network parts, while keeping the ability to decrypt all traffic in an out-of-band-system without knowledge of specific cryptographic state details. By conducting a performance evaluation, we highlight the benefits of *LLPR-CS* and discuss its advantages compared to existing approaches.

Keywords: ISO 11783; ISOBUS; Controller Area Network; Security; Privacy; Confidentiality; Smart Farming

1 Introduction

The CAN system is extensively used in many kind of modern automotive and industrial control systems. Due to its wide adoption and versatile applications, there exist many reasons to use this system even though achievable data rates are heavily limited. From a perspective of application developers, one of the main reasons is – beside the good reliability – the simplicity of CANs, which allows to implement all networking functions on the application layer. This advantage lies in the fact that CANs form a simple broadcast-only topology, so that no routing or expensive network coordination is required. However, this topology has important implications for security in such systems. While the security of traditional, air-gapped systems could basically rely on the physical inaccessibility of the network, for current CANs, it is no longer possible to solely rely on this implicit security measure, since the industrial demand for the vertical integration into the Internet of Things (IoT) is very common in many recent developments in areas like the Industry 4.0 or Smart Farming.

[1] University of Osnabrück, Institute of Computer Science, Wachsbleiche 27, 49076 Osnabrück, Germany tzimmermann@uos.de
[2] University of Osnabrück, Institute of Computer Science, Wachsbleiche 27, 49076 Osnabrück, Germany bauer@uos.de
[3] University of Osnabrück, Institute of Computer Science, Wachsbleiche 27, 49076 Osnabrück, Germany aschenbruck@uos.de

Whereas the most obvious security issue, the inability to verify the integrity and authenticity of received messages, was discussed in multiple papers [HKD08; KY17; Wa17], the confidentiality of message delivery still remains challenging in real-world scenarios. To this end, we propose a simple and lightweight approach to seamlessly retrofit encryption into existing systems. This approach preserves the length information of original payload and, thus, increase the interoperability, particularly with higher-layer protocols. Moreover, a suitable block cipher encryption is adapted that offers a secure and robust confidentiality.

The paper is organized as follows. Our motivational scenario and background details on CAN and security threats are given in Sec. 2, followed by a brief discussion on related work (Sec.3). Then, in Sec. 4, four different cryptographic variants are introduced including *LLPR-CS* and related approaches. Finally, the performance evaluation is presented in Sec. 5 and Sec. 6 concludes the paper.

2 Background

In this paper, the ISOBUS is exemplarily used. This protocol interconnects agricultural machinery to provide high-resolution control, e.g., of seeding and fertilization. Often heterogenous actors are involved, regarding both technical (machinery) and organizational (contractors) aspects. In agricultural practice, on the one hand, it is common to rely on contractors for specific work in the field. These contractors can either use their own or customers' machines. On the other hand, due to the high cost and high specialization, one may rent so-called *implements*, which may be third party trailers, planters, or sprayers.

The agricultural environment with different actors has neglected privacy aspects for long periods. However, due to the digitalization and networking using farm management information systems (FMISs) that are often cloud-based, these aspect become highly relevant in todays agriculture. High-resolution data is required to release the potential of Precision Agriculture and cooperation between actors, manufacturers, and implement owners may use their ability to get own hard- and software in the consumer's system. Therefore, different parties can record, interpret, and (ab-)use data collected by machinery. This data could reveal business data, e.g., soil or yield properties of individual fields, or personal data of employees.

2.1 Controller Area Network (CAN) in Precision Agriculture

CAN (ISO 11898) [ISO15; Ok05] is a simple and robust broadcast-only bus system using differential signaling. There exist different generations of the CAN protocol. The version CAN 2.0A has a frame format consisting of an 11 bit identifier, up to 8 bytes for payload, and several internal fields, such as Cyclic Redundancy Check (CRC) and Data Length Code (DLC). The identifier is used as an arbitration field allowing to avoid collisions, using

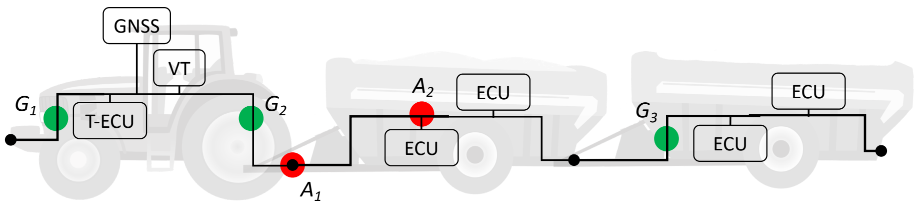


Fig. 1: ISOBUS network with encrypting gateways. (GNSS: global navigation sattelite system receiver).

a physically supported listen-while-talk multiple access scheme, i.e., Carrier Sense Multiple Access/Collision Resolution (CSMA/CR). CAN 2.0B enlarges the identifier field to 29 bit, allowing a total of 93 bits user-defined data to be transferred in a frame.

ISOBUS (ISO 11783) [ISO07] is a protocol based on CAN that additionally defines a full application protocol. It is specified for CAN 2.0B with a fixed data rate of 250 kbit/s. ISOBUS applications reuse the identifier field of CAN frames to transfer message headers by splitting it into multiple fields. Some parts of the message header are used to allow for backwards compatibility with existing protocols and also for message prioritization (based on the arbitration field of the original CAN). The most notable part is the so-called Parameter Group Number (PGN), which, in conjunction with a publicly available database of PGNs[1], enables a bit-wise interpretation of the payload. This allows flexible applications, such as the configuration of a Virtual Terminal (VT) in the tractor's cabin to be shared as central GUI for all implements connected to the tractors' ISOBUS (cf. Fig. 1).

2.2 Scenario & Security Threats

The motivating scenario which was already sketched above is shown in Fig. 1. Possible access points for malicious data collection are ISOBUS connectors between machines and implements that are exemplarily represented by A_1 and A_2. Due to the broadcast medium, a potential attacker could unnoticeably add a logging device to those points with minimum effort. By adding gateways encrypting all private messages ($G_{1,2}$) and its counterpart (G_3) inside trusted implements, these points are no longer useful for malicious actors. This architecture, however, assumes that all units connecting sensors and actors to the bus (called Electronic Control Unit (ECU) in automotive terms) inside both, the tractor as well as the (trusted) second implement, are under control of the data owners. Despite of being practically relevant, this exemplary agricultural scenario is not the only one highly requiring confidentiality. The encryption of certain data prior to the cloud upload is also conceivable, for instance.

[1] `https://www.isobus.net/isobus/pGNAndSPN`

3 Related Work

Over the recent years, high effort has been put into researching the security aspects of CANs. However, most of these works were focused on message integrity and authenticity, which are – in terms of vehicle networks – mainly necessary to ensure operational safety. Both is vital to protect against physical harms of drivers or the vehicle itself, yet it does not fit well in scenarios in which the direct, physical harm is not the goal of an attacker. Hoppe et al. summarized this considerations in the commonly used terms of cybersecurity aspects (cf. the CIA rule [NIST95]). They declare the reason for the inability to expect integrity and authenticity as the lack of sender identification, in combination with the shared-medium, broadcast nature of CANs [HKD08]. The aspect of data availability is additionally endangered due to the internal priority mechanisms, which could allow an attacker to continuously send high-priority messages preventing any useful communication on the bus. Since the safety issues introduced by missing authentication and integrity mechanisms are practically relevant, Wu et al. [Wu16] present an approach that uses Hashed Message Authentication Codes (HMACs) in CAN. HMACs necessarily require an inherent overhead (additional data to be transmitted), which cannot be realized in conventional CAN messages with a maximum payload size of 8 bytes. Therefore, the approach relies on payload compression in order to achieve sufficient room for HMAC overhead in the payload.

The confidentiality requirement is addressed by Bruton [Br14], who proposes different encryption schemes. To use widely adopted encryption algorithms, especially the Advanced Encryption Standard (AES), it is necessary to implement an adequate segmentation of the encrypted payload, as encrypted AES blocks are 128 bit in length and, thus, do not fit into a single CAN frame. However, segmentation will inevitably impose a significantly increased bus load and drastically reduce the goodput. Hence, RC4 stream cipher is demonstrated as an alternative [Br14]. Beside RC4 is no longer recommended, stream ciphers are generally prone to desynchronization, because even an offset of a single byte in the key stream renders all transferred data to be broken. Thus, they need an additional synchronization mechanism. Moreover, data enciphered with stream ciphers cannot be deciphered afterwards without storing the key stream position and other state information, such as the used initialization vector (IV). In a short demonstration, Jukl and Čupera [JČ16] present a similar approach to encrypt ISOBUS messages using the Tiny Encryption Algorithm (TEA). This algorithm features a 64 bit block size and, therefore, does not need a segmentation when being applied to CANs. Their work is mainly focused on the ability to execute the encryption on a commercially available microcontroller platform. Furthermore, solely ISOBUS communication is considered. This makes their approach very specific and not applicable to pure CAN systems of other domains. Overall, in the current literature, there is a lack of approaches for a feasible confidentiality solutions that can be applied to existing CANs with minimal modifications of existing systems in terms of hardware, software, and bus load.

4 Adding Confidentiality to CANs

4.1 System Requirements

In order to meet different requirements, it is our goal that an encrypted communication should be usable in two architectures: The most obvious solution is to insert stand-alone gateways G_i as proxys into an existing network to realize encrypted tunnels as shown in Fig. 1, where the whole section between G_2 and G_3 could be encrypted, preventing malicious attackers from reading data either at the physically accessible port A_1 or an integrated ECU (A_2). This may especially be effective in environments where clearly defined network sections in the topology exists, in which data should be encrypted, but existing devices cannot be updated or replaced. For a clear modularization, in our approach, each encryption scheme features two software components, one for encryption and one for decryption. In this paper, the encryption key is expected to be present on both the en- and decryption side. The actual key exchange is out of scope but could be realized by corresponding key exchange protocols.

4.2 Cryptographic Variants

Because there are varying system requirements for different use cases, multiple encryption variants were developed and implemented in this work. These include block and stream cipher schemes. Even though all described schemes can generally be used in CANs, every variant has its own limitations, advantages, and disadvantages, which are summarized in Tab. 1 and discussed in the following.

4.2.1 Block Cipher

***Bruton* (AES)** To verify the usability and to have a homogenous measuring environment, the proposed solution from [Br14] using AES encryption was implemented in our software framework. Each received CAN frames payload is encrypted with a pre-shared key using the Cipher Block Chaining (CBC) operation mode, which is strictly necessary to prevent attacks based on differential cryptanalysis. As the payload only consists of maximum 64 bits and AES block size is always 128 bit, the additional bits are used to encode the length

	Variant	Encryption	Length preservation	Network overhead	Fault resistance	Practical applicability	Cryptogr. security
🟪	*Bruton*	AES	X	–	–	+	+
🟩	*Inline-ISOBUS*	XTEA	X	○	+	–	○
🟦	*LLPR-CS*	XTEA	✓	○	+	+	○
🟥	*Stream Cipher*	ChaCha20	✓	+	–	○	+

Tab. 1: Advantages (+) and drawbacks (–) of different encryption variants.

of the received payload. This additional information is necessary to send the decrypted message with its original length and without padding as upper layer protocols may rely on the message's length as part of their protocol. Unused bits are simply set to zero, as the knowledge of partial bits in a single block doesn't have any security implications. Once encrypted, the resulting block is split into two message segments that are transmitted using the original identifier. The receiving side simply decides how to handle even and odd packets, and resembles both segments. After decrypting such a resembled frame, it gets sent to the unencrypted bus segment or unmodified software expecting unencrypted data. Obviously, one can safely rely on *cryptographic security* of this method, since AES is considered as state-of-the art and used in nearly every modern encryption system. However, this variant is expected to have a highly negative impact on the *bus load*, as every unencrypted frame needs to be sent in two different frames (with maximum payload size) containing a part of the ciphertext.

Inline-ISOBUS To prevent excessive overhead and to prevent degradation of fault tolerance as present in the AES-method, it is possible to choose a 64 bit block cipher [JČ16]. The use of TEA is one possibility, while the underlaying block cipher can be chosen freely depending on known attacks against any of them. By using TEA as demonstrated in [JČ16] in CBC mode, the design is fairly simply, requiring only knowledge of the previous block and an IV or – given that single incorrect transmission do not present a problem for the application – even without this. The biggest advantage of the CBC mode is its self-synchronizing functionality, i.e., there is no counter involved and the decryption for block n only relies on the ciphertext of block $n-1$. While the use of 64 bit block ciphers is generally recommended against because of common attack vectors when used in conjunction with the CBC mode [BL16], this potential vulnerability is very unlikely to present a problem in CAN-based systems, as the data rate is strictly limited. Hence, it is nearly impossible for an attacker to capture the necessary large amount of cipher text. While this solution does not require any state information to decrypt captured traffic and, therefore, can be used in heterogenous systems as well, it still shows a severe limitation compared to the AES method. Because the encryption is a strictly bijective transformation for the whole payload of 64 bit and cannot be used for partial blocks, it is impossible to encrypt blocks shorter than 64 bit. To encrypt messages smaller than 8 byte, they are padded with arbitrary values to a full 8 byte-block. As traditional padding schemes which allow for the removal of the padding before transferring the payload to the upper layer protocol are built for messages with multiple consecutive blocks, these are not applicable in CANs. Thus, this variant is unable to *preserve the length* of the unencrypted message. Hence, suchlike encryption schemes can only be used if the upper layer protocol ignores additional data, which is the case in ISOBUS, but cannot be necessarily expected for other CAN-based protocols.

LLPR-CS To overcome this limitation of missing length preservation, a few bits for additional metadata are sufficient that simply specify the number of padding bytes used. The DLC field in the CAN frame header allows for that few extra bits of information to be encoded in a standard-compatible way, without breaking compatibility to existing hardware.

As only 9 of the 16 available states representable by 4 bit have a valid meaning, the CAN standard declares all other states to be interpreted as a 8 byte payload [ISO07, Part 4]. Therefore, the unused states can be safely used to indicate the number of padding bytes which have to be removed by the decrypting side. The case of a zero-byte payload cannot be signaled this way, however it is feasible to simply send an empty frame in this case, because no data content is available for encryption.

4.2.2 Stream Cipher

The most relevant challenge for using block ciphers in CAN-based networks is the fixed block size, which either leads to the need for fragmentation, or any form of padding with the shown difficulty of discarding the original length. To overcome these problems, it may be promising to use stream ciphers instead. They allow for encryption of any chunk of data with arbitrary length, as the encryption consists only of the bitwise addition with an independent key stream. Accordingly, this wouldn't require the use of any padding, and, therefore, can preserve the length of transmitted frames. In some situations however it may be useful to record messages sent on the CAN, especially in agricultural use cases this is done to allow for offline analysis of collected data. As every message encrypted using a stream cipher needs to be decrypted with the correct portion of the key stream, it would be necessary to store the messages' position in the encryption session. Furthermore, even when this should not happen in simple bus scenarios, a single lost message invalidates the counter for the specific entity, this variant results in a very limited *fault tolerance*, as only reordered messages may happen. This makes it necessary to add an additional mechanism for synchronizing the counter.

5 Evaluation

5.1 Implementation

For our proof-of-concept evaluation, a fully-featured single board computer is used in this paper, which allows for a more flexible development system. The introduced encryption variants were prototypically implemented on a Raspberry Pi and were integrated in the open-source *CAN't* [Ba19] framework[2]. All variants were implemented in stand-alone binaries to allow for easy adoption to existing software using SocketCAN. Publicly available and maintained libraries were used whenever available, namely *OpenSSL* for the AES-based variant (*Bruton*), and *libsodium* for the ChaCha20 algorithm (*Stream Cipher*). For the implementation based on TEA, the reference implementation of the corrected and eXtended TEA from the authors was used [WN98].

[2] `https://sys.cs.uos.de/cant`

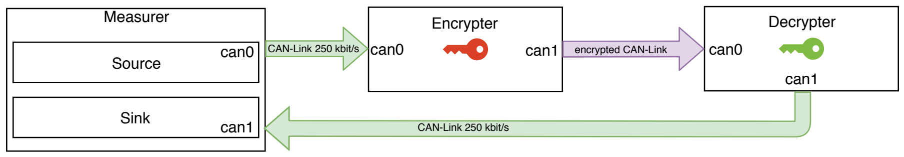


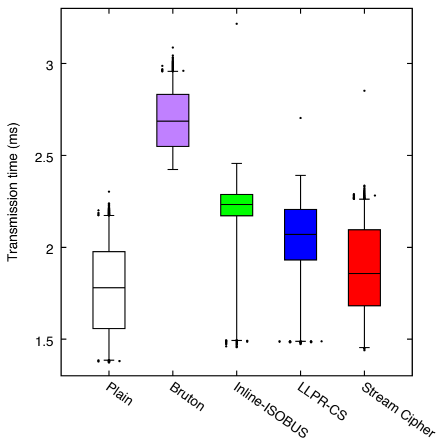
Fig. 2: Evaluation setup.

5.2 Methodology & Evaluation Setup

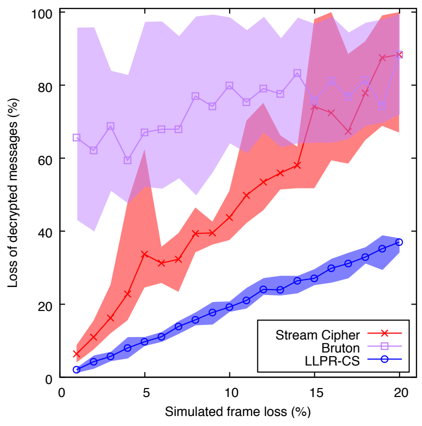
To achieve a deeper insight on the impact of the different encryption schemes on a real system, we set up a testbed consisting of four logical communication units as shown in Fig. 2. Two of them form a pair of legitimate bus nodes (*Source* and *Sink*), which need to communicate in a confidential manner. The other two units serve as an *Encrypter* and a *Decrypter*, establishing an additional and encrypted bus segment that could be publicly exposed. This setup represents the worst case in terms of delay, because the single bus segment of the original setup (without the encrypted tunnel segment) is divided into three independent CANs. Therefore, the time frames spend on the communication medium is expected to get at least tripled. Because *Source* and *Sink* are integrated in a single physical node (*Measurer*, i.e., a Raspberry Pi with two CAN interfaces), precise delay measurements are possible without potential time synchronization inaccuracies. As described in Sec. 4.2.1, all block ciphers were used in CBC mode.

5.3 Results

The **transmission delay** induced by our method is measured by the duration for the successful transmission of random payloads with uniform distributed, random lengths from *Source* to *Sink*. Even though every application may have specific limits on maximum message delays, the additional delays the encryption adds are fairly low as shown by the boxplots in Fig. 3(a) (10 000 frames for each variant). The most obvious delay factor may be the encryption itself, which was evaluated in an additional measurement. However, as all encryption algorithms are developed to allow fast encryption and decryption of large data, the differences between them are mostly negligible to the differences resulting of larger or additional data transfers. Therefore, and because there are many benchmarks for cryptographic algorithms publicly available, these results are left out here. As expected, the *Bruton* variant that splits one frame into multiple frames is obviously the slowest one. It increases the mean delay from 1.77 ms to 2.69 ms. According to [ISO07], the encrypting gateway is to be handled as a network interconnection unit (NIU). The standard recommends a maximum processing time of 10 ms, which is found to be uncritical for all variants. Nonetheless, the novel *LLPR-CS* variant enables a significant lower delay than *Bruton*'s variant and, due to its length preserving, it allows to remove padding bytes after decryption which leads to a lower transmission delay than *Inline-ISOBUS*. Thus, it is feasible in delay-critical systems while preserving the advantages of block encryption schemes over stream ciphers.



(a) Transmission delays from *Source* to *Sink*.



(b) Loss sensitivity (min, max, mean for each variant).

Fig. 3: Testbed evaluation results of different encryption schemes.

In many scenarios, it may be vital to keep the **bus load** as low as possible, because of CANs limited throughput. Similar to the transmission delay, we also investigated the additional bus load induced by each variant, but do not present details here due to space limitations. However, expected results were found. It is obvious that *Bruton*'s AES solution is only suitable in environments with a prior load below 50% capacity since the fragmentation doubles the load. Whereas the stream cipher variant has no impact on bus load, the overhead of both variants, *LLPR-CS* and *Inline-ISOBUS*, depends directly on the typical length of transmitted messages, as their overhead is exactly the padding required to form 8-bit-blocks.

So far, all measurements were carried out in a fully reliable CAN. This may, of course, not always apply. Thus, it is necessary to consider the effects on transfer **robustness** of all variants. Even though CAN is definitely focused on preventing frame loss and itself guarantees an in-order transmission of messages, this assumption may be invalidated as higher priority messages prevent the transmission of lower priority ones in nearly saturated networks. Additionally, in very noisy environments or near-fault conditions, this expectation may no longer hold true and single frames can be lost. This is especially relevant when any kind of gateway (cf. NIU as specified in ISOBUS) is part of a network, as these gateways may reorder messages during the forwarding process. By developing a simple loss emulation tool, it was possible to measure the impact of CAN-frame loss with a specified probability on the fully encrypted channel. All measurements were carried out with 1000 messages and 10 replications. In Fig. 3(b), the results concerning the impact of frame loss are shown for all variants (except *Inline-ISOBUS* that is identical to *LLPR-CS* here). Here the *loss of decrypted messages* is the percentage of messages, which either are never delivered or received with a corrupted payload. For both variants of 64 bit-block ciphers, the resulting loss of correct messages behaves proportional to the loss of underlying CAN messages. This matches the expectations. If the payload of a certain frame is lost, the next frame cannot be correctly decrypted. However, the second following frame can be decrypted correctly. The reason for the factor not being exactly two is the fact that when n consecutive frames get lost in a burst, only $n + 1$ messages are lost or incorrectly decrypted. To evaluate the stream-cipher solution, it was necessary to extend the system by a minimal synchronization method. Without this, all messages transmitted after the first loss occurred would be considered lost.

Overall, the evaluation showed a wide variety of advantages and disadvantages, both in terms of performance characteristics as well as provided features and requirements of the implemented variants. The selection of one of these variants, therefore, relies on a deep knowledge of the characteristics for an existing system. Based on differentiating between them, it is possible to give rough suggestion which variant may be useful. Whenever it is possible to save additional metadata like algorithm's state and the reliability of the CAN is ensured, using a stream cipher can be recommended, as it has only minimal overhead due to its fast processing time. If these prerequisites are not met, the upper layer protocol has to be considered. Specially in ISOBUS-networks, any 64 bit block cipher can be used directly. However, it is important to limit the maximum of messages sent with a single key to circumvent problems in using the CBC mode.

To use block ciphers for generic upper layer protocol, our proposed LLPR-CS allows to do this in a backward-compatible manner without violating the CAN standard by reusing the DLC-bits to mark the padding's length. A small drawback of this solution is the handling of frames received by unmodified gateways whose may discard the additional bits used to indicate the padding. Nonetheless, this solution is lightweight, as it requires no extra storage of state information and robustness by the underlaying block cipher. Additionally, with our solution it is possible to establish a completely transparent encrypted channel, as it preserves the length of transmitted frames.

6 Conclusion

In this paper, we propose *LLPR-CS*, an efficient and interoperable confidentiality solution that leverages unused bits in the header of CAN frames. Although our work is focused on ISOBUS of agricultural machines, the presented solution is applicable to general CAN systems beyond Smart Farming. With prototypical implementations, we compared our novel solution against related approaches. The performance evaluation shows the benefits of *LLPR-CS* with regard to the additional transmission delay and frame loss sensitivity. However, due to space limitations, both, the low impact on additional bus load and its robustness against frame disordering, could not been presented here. Moreover, we also implemented a feature that allows to selectively encrypt ISOBUS messaged according to their PGN, which was also not explicitly addressed in this paper. In our future work, we plan to evaluate our solution on real agricultural machines. Here, the feature of selective encryption is expected to become more crucial to meet the confidentiality requirements, particularly from a privacy perspective, without interrupting the operation of the machinery.

Acknowledgments

This work was partially funded by the German Federal Ministry of Education and Research (BMBF) within the Program “Innovations for Tomorrow’s Production, Services, and Work” (02K14A194) and managed by the Project Management Agency Karlsruhe (PTKA). The authors are responsible for the contents of this publication.

References

[Ba19] Bauer, J.; Helmke, R.; Bothe, A.; Aschenbruck, N.: CAN’t track us: Adaptable Privacy for ISOBUS Controller Area Networks. Computer Standards & Interfaces 66/, DOI: 10.1016/j.csi.2019.04.003, 103344:1–103344:13, 2019.

[BL16] Bhargavan, K.; Leurent, G.: On the Practical (In-)Security of 64-bit Block Ciphers: Collision Attacks on HTTP over TLS and OpenVPN. In: Proc. of the Conf. on Computer and Communications Security (SIGSAC). CCS ’16, DOI: 10.1145/2976749.2978423, Vienna, Austria, pp. 456–467, 2016.

[Br14] Bruton, J. A.: Securing CAN Bus Communication: An Analysis of Cryptographic Approaches, MA thesis, National University of Ireland, Galway, Aug. 2014.

[HKD08] Hoppe, T.; Kiltz, S.; Dittmann, J.: Security Threats to Automotive CAN Networks – Practical Examples and Selected Short-Term Countermeasures. In (Harrison, M. D.; Sujan, M.-A., eds.): Computer Safety, Reliability, and Security. Springer, Berlin, Heidelberg, pp. 235–248, Sept. 2008, ISBN: 978-3-540-87698-4.

[ISO07] International Organization for Standardization (ISO): Tractors and machinery for agriculture and forestry – Serial control and communications data network – Parts 1–14, ISO 11783-{1–14}:2007–17, 2007.

[ISO15] International Organization for Standardization (ISO): Road vehicles - Controller area network (CAN) – Part 1: Data link layer and physical signalling, ISO 11898-1:2015, 2015.

[JČ16] Jukl, M.; Čupera, J.: Using of tiny encryption algorithm in CAN-Bus communication. In: Research in Agricultural Engineering. Vol. 62, DOI: 10.17221/12/2015-RAE, pp. 50–55, June 2016.

[KY17] King, Z.; Yu, S.: Investigating and securing communications in the Controller Area Network (CAN). In: Proc. of the Int. Conf. on Computing, Networking and Communications (ICNC). DOI: 10.1109/ICCNC.2017.7876236, Santa Clara, CA, USA, pp. 814–818, 2017.

[NIST95] National Institute of Standards and Technology: An Introduction to Computer Security: The NIST Handbook. Special Publication 800-12, DOI: 10.6028/NIST.SP.800-12r1, Oct. 1995.

[Ok05] Oksanen, T.; Öhman, M.; Miettinen, M.; Visala, A.: ISO 11783 – Standard and its Implementation. IFAC Proceedings Volumes 38/1, pp. 69–74, 2005.

[Wa17] Wang, E.; Xu, W.; Sastry, S.; Liu, S.; Zeng, K.: Hardware Module-Based Message Authentication in Intra-vehicle Networks. In: Proc. of the Int. Conf. on Cyber-Physical Systems (ICCPS). Pittsburgh, PA, USA, pp. 207–216, 2017.

[WN98] Wheeler, D. J.; Needham, R. M.: Correction to xtea, tech. rep., Computer Laboratory - Cambridge University, Oct. 1998, URL: `https://www.movable-type.co.uk/scripts/xxtea.pdf`, visited on: 09/30/2019.

[Wu16] Wu, Y.; Kim, Y.-J.; Piao, Z.; Chung, J.; Kim, Y.-E.: Security protocol for controller area network using ECANDC compression algorithm. In: Proc. of the Int. Conf. on Signal Processing, Communications and Computing (ICSPCC). DOI: 10.1109/ICSPCC.2016.7753631, Hong Kong, China, pp. 1–4, 2016.