SENECAN: Secure KEy DistributioN OvEr CAN Through Watermarking and Jamming

Simone Soderi, *Senior Member, IEEE*, Riccardo Colelli, Federico Turrin, Federica Pascucci, *Senior Member, IEEE*, and Mauro Conti, *Fellow, IEEE*

Abstract—The Control Area Network (CAN) represents the standard bus for intra-vehicular networks communication. Unfortunately, CAN was not designed to be a secure protocol. Communications over CAN do not take advantage of any security feature (e.g., cryptography and authentication), raising different vulnerabilities in critical applications. This lack of security is even more emphasized in recent CAN networks, which integrate remote connection capabilities (e.g., Bluetooth and WiFi). This insecurity-by-design led to the development of specific mechanisms to patch CAN vulnerabilities. Many proposed solutions rely on implementing optimized cryptographic primitives and assume that the cryptographic keys were previously shared among the different nodes during the production phase, omitting the issue related to keys distribution and update. We propose SENECAN, a solution that combines watermarking and wired jamming to secure the CAN bus's long-term key distribution. Our solution leverages intentional interference and spread spectrum watermarking to achieve security properties such as confidentiality, integrity, authentication, and anti-replay. Compared to other works, SENECAN does not require any CAN protocol and system architecture modification. Instead, it requires an additional CAN transceiver and an initial transmission overhead. Finally, we tested the effectiveness and functioning of the SENECAN distribution schema in a real CAN environment.

Index Terms—Controller area network (CAN), security, key distribution, jamming, watermarking, vehicular network

1 Introduction

The increase of interconnection and automation in automotive systems led to the development of new connection and communication standards among their components. The Control Area Network (CAN), introduced in 1993, is the most common standard used for Intra-Vehicular Networks (IVNs) communication and other applications such as railways and industrial automation. In a vehicular system, the network's nodes are called Electronic Control Units (ECU). These nodes communicate in broadcast through the CAN bus. Each ECU represents a single function in the machine, such as the engine control unit, airbags, or audio system. Modern vehicles include up to 200 ECUs, which continuously exchange messages to monitor the car's behavior. Due to the continuous integration of new services in vehicles, the number of ECU will significantly increase.

CAN bus was initially created without considering security issues and was designed to operate in isolated LANs without any external interaction. In particular, CAN communication lacks both encryption and authentication mechanisms, making it sensible to integrity, authenticity, and confidentiality attacks. Over the years, such lack of security properties was exploited by researchers to demonstrate the feasibility of spoofing attacks [1], command injection [2], eavesdropping [3], and replay attacks [4]. Despite that, in the last years, cars have been connected to external services such as GPS navigation systems, 5G connections, and Bluetooth. This open new vulnerability surfaces exploitable by hackers, also facilitated by the CAN bus's inherent vulnerability. One of the most famous events that attracted public attention on vehicle security happened in 2015 when two hackers showed that they could hack a Jeep [5] remotely. However, more recent studies also highlighted the exploitation of CAN bus with other attacks such as injection from outside the IVN via wireless channel [6] or ransomware injection through Over-The-Air updates (OTA) [7]. In 2019, the National Security of Standards and Technology (NIST) issued the Special Publication 800-160 [8], which describes techniques and approaches to improve the cyber-resiliency of systems. NIST also considered self-driving cars as a system based on new emerging technologies among the examined use cases. The analysis showed that an adversary could subvert autonomous technology to divert and potentially crash the vehicle. Indeed, in the hypothesized threat scenario, the adversary-installed malware data, and commands onto the CAN bus, gaining the remote control of the network. One of the possible mitigations is validating data sent and received among nodes of the network.

To guarantee the system's safety and consequently people's safety, new security mechanisms to protect CAN communication must be designed. The majority of the proposed solutions rely on modifying the CAN standard (e.g., cryptographic primitives [9], [10]) or integrating additional nodes (e.g., Intrusion Detection Systems [11], [12], [13]). However,

- *Simone Soderi is with the IMT School for Advanced Studies Lucca, 55100 Lucca, Italy. E-mail: simone.soderi@imtlucca.it.*
- *Riccardo Colelli and Federica Pascucci are with the Department of Engineering, University of Roma Tre, 00146 Rome, Italy. E-mail: {riccardo.colelli, federica.pascucci}@uniroma3.it.*
- *Federico Turrin and Mauro Conti are with the Department of Mathematics, University of Padua, 35121 Padua, Italy. E-mail: {turrin, conti}@math.unipd.it.*

Manuscript received 27 October 2021; revised 20 May 2022; accepted 27 May 2022. Date of publication 1 June 2022; date of current version 13 May 2023. (Corresponding author: Simone Soderi.)
Digital Object Identifier no. 10.1109/TDSC.2022.3179562

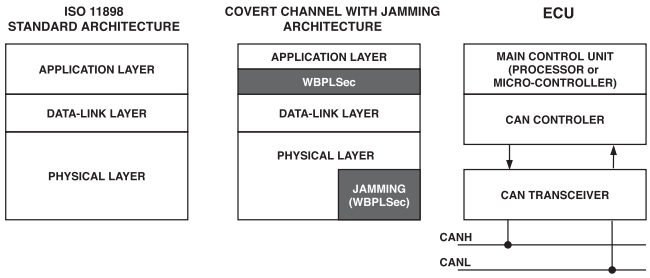


Fig. 1. Integration of WBPLSec in the CAN protocol.

such solutions are sometimes impossible to implement due to market constraints and are difficult to integrate into current systems.

A common solution to hide information in the communication is represented by covert-channels, introduced by Lampson in 1993 [14]. This technique exploits unintended channels to transmit information. In other words, in covert-channels specific data is embedded in a transmission medium that is not supposed to transfer such data. In literature, this technique has been implemented to exfiltrate information from networks without violating the expected behavior, thus without triggering Intrusion Detection Systems (IDS) (e.g., in [15], [16]). In recent years different works have employed covert-channel to implement security mechanisms, e.g., authentication [17], [18], [19], [20]. The strength of this approach is that by stealthy inserting the information within an existing medium, there is no need to modify the existing protocol.

Contribution. In this paper, we propose an authenticated key distribution system for CAN communication based on a combination of a watermark-based technique and jamming over a wired network. Indeed, most of the related work assume the existence of a long-term secret shared among the different ECUs installed by the manufacturer. We propose and validate a schema to dynamically and securely share or refresh this secret between the ECUs by considering the CAN network communication constraints. Similarly to [21], we define jamming over CAN bus the *intentional interference* operation on the signal aimed at destroying the communication between devices. CAN communication presents broadcast messages on a shared medium, and therefore jamming can block reception of the message for all the listening nodes. Unlike previous works, our mechanism does not require any additional modification of the CAN standard or the CAN network architecture (e.g., adding new nodes). Our key distribution system exploits the traditional bit collision mechanism at the physical level to destroy part of the shared secret and reconstruct it at the receiver side. To do this, we use the Watermark Blind Physical Layer Security (WBPLSec) protocol which utilizes a jamming receiver in conjunction with a Spread-Spectrum (SS) watermarking technique [22]. The CAN specification (i.e., ISO 11898) considers three levels of the OSI model: physical, data-link and application layers. Fig. 1 represents the standard layer according to the CAN specification and shows how we integrated the WBPLSec algorithm with it. The architecture we propose can be considered a Bump-In-The-Stack (BITS) [23], where *watermarking* and *jamming* are two atomic functions operating within the standard CAN stack. Instead, the architecture of the standard ECU is not modified; indeed, WBPLSec can be implemented at the software level in the ECU micro-controller to perform the watermarking while the CAN transceiver is used to perform the jamming on the bus. By using this method to secure the communication, the original message to be transmitted is passed from the application layer (i.e., ECU micro-controller) to the WBPLSec function, which embeds an SS watermark in the information before passing it to the lower layers. More precisely, we use watermarking as a covert-channel. Instead, on the receiver side, the node selectively jams the transmitted message on the CAN bus using the jamming function added in the stack, making part of the communication unusable for the attacker.

As a result, our solution can be easily implemented in the existing CAN system by installing a transceiver on the ECUs.

We summarize our contributions as follows:

- We present SENECAN, an innovative mechanism that exploits jamming at the physical layer over wired communications to implement a secure key distribution phase.
- We implement and test SENECAN on a CAN bus environment proving its effectiveness. SENECAN combines watermarking and jamming and guarantees confidentiality, integrity, authentication, and anti-replay capabilities without modifying the protocol architecture.
- We survey related works' strengths and limitations regarding security properties and requirements and compare them with SENECAN. Compared to other works, SENECAN implements more robust security properties without requiring additional nodes.

WBPLSec algorithm was already applied with success on different communication means such as radio frequency [22], acoustic communications [24] and visible light communications [25]. To the best of our knowledge, this is the first work combining watermark-based communication with jamming over a wired network to implement a key distribution mechanism over the CAN network.

Organization. The remainder of the paper is organized as follows. Section 2 briefly recalls the main concepts useful for the goal of the paper, while Section 3 provides an overview of the related work. Section 4 outlines our system model assumption. Sections 5 and 6 illustrate the proposed key exchange mechanism and the experimental validation on real equipment, respectively. Section 7 discusses the security properties of SENECAN. Section 8 examines the results and compares them with the other schemes in the literature. Finally, Section 9 concludes the paper.

2 Background

In this section, we briefly recall the main concepts helpful in understanding the remainder of the paper. In particular, in Section 2.1 we recall the CAN protocol, with a specific focus on the physical layer properties, which we exploit to perform the jamming. Then, in Section 2.2 we present the application of covert-channels and Watermarking techniques in the security field.

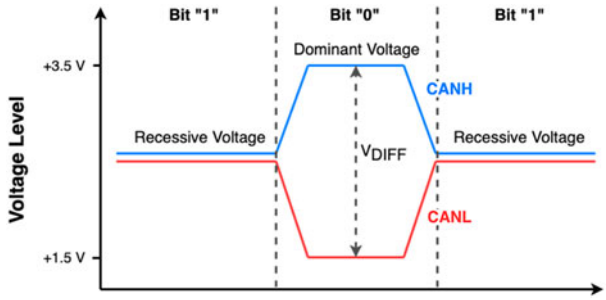


Fig. 2. Collision management in CAN protocol (ISO 11898).

2.1 Controller Area Network (CAN)

CAN bus is the most common and widely diffused medium for in-vehicular communication. CAN was initially released in 1986 by the Society of Automotive Engineers. However, nowadays is also implemented in other applications such as Building Automation and Transportation applications.

The CAN standard is defined in the ISO 11898, initially released in 1993 [26]. In particular, the physical layer characteristics of CAN are defined in the ISO 11898-2 [27], also known as *high-speed* CAN, and ISO 11898-3 [28], also known as *low-speed* CAN or *fault-tolerant* CAN. CAN enables communication among different nodes or Electronic Control Units (ECUs) on a differential bus. The communications rely on information broadcast, using a linear bus, star bus, or multiple star buses connected by a linear bus, terminated at each node by a resistance of 120 Ω. The base CAN frame consists of 108 bits, while the extended version of the can protocol has a length of up to 128 bits. Since the CAN protocol uses a broadcast transmission, every node receives every frame. To specify the receiver of a frame, there are two standard specifications. The CAN 2.0A defines 11 bit of identifier field, while the CAN 2.0B defines 29 bit of identifier. In this work, we consider the CAN 2.0A specification. The data field can contain up to 64 bits of information. Finally, the 16 bit CRC field identifies transmission errors and represents the protocol's only security feature. Fig. 2 illustrates the CAN physical layer transmission model, which relies on dominant signals, or CAN High (CANH) (encoded as logical *0 s*) and recessive signals, CAN Low (CANL) (encoded as logical *1 s*). An ECU comprises three elements: a processor, the CAN controller, and the CAN transceiver. According to ISO 11898-1 [29], which defines CAN data-link layer and physical signaling, the CAN controller and the CAN transceiver components of each ECU manage the transmission of data along the CAN bus. In particular, the CAN controller is responsible for assembling the frame at the data-link layer, while the CAN transceiver translated logical signals received from the CAN controller to the physical voltage level and vice versa. To avoid the frame collisions due to the broadcast transmissions and the shared medium, the CAN protocol implement a Carrier Sense Multiple Access/Collision Resolution (CSMA/CR) mechanism based on ID priority: the frame with the highest priority (lower arbitration ID) gain access to the bus, while all other nodes (higher priority arbitration ID) switch to a "listening" mode. Furthermore, if two nodes transmit a frame with the same priority level, if one node transmits a dominant bit and another transmits a recessive bit, in the resulting collision, the dominant bit "wins" between the two (i.e., logical AND combination). We leverage this collision mechanism to perform intentional jamming at the physical layer level. This will be detailed in Section 4.1.

2.2 Covert-Channels and Watermarking

Physical Layer Security (PLS) aims at securing communications by exploiting the physical properties of the communication channel. These techniques include processing the signal sent over a channel to obtain specific security properties without resorting to protocols or algorithms at upper layers than the physical one.

Protecting data from unauthorized access is indeed a fundamental aspect of communications. Cryptographic protocols answer this need. Only those authorized to participate in the communications have the credentials to interpret the protocol. Unfortunately, there are cases in which encryption is insufficient, and unconventional communication channels might solve the problem. In 1973, Lampson defined a *covert-channel* as a communication channel that is not intended for information transfer at all [14]. In literature, several contributions investigate how to implement effective covert-channels in different layers of the network. For instance, Hanspach *et al.* proposed the utilization of covert-channels to circumvent network security policies by establishing new communication paths [30]. Moreover, we can use hidden channels for various legitimate and non-legitimate purposes. While governments and companies can use these channels to protect their communications, cyber-criminals can also exploit this technology in the same way. In other words, covert-channels are just another way for digital communications through information hiding [31].

Watermarking is a technique used to hide or embed a signal into another signal, e.g., pictures and videos. PLS can be implemented with spread-spectrum watermarking techniques [32]. In particular, we implement the first equation described by Cox *et al.* [33]. We embed information in the original message through an SS watermark.

Essentially, the SS watermark information is overlapped with the original information stream and travels within the CAN protocol frames. According to Lampson [14], a communication channel made in this way is a covert channel. Like other types of covert-channels, watermarking-based covert-channels allow embedding the additional information without using any additional communication channel. In our case, this specific watermark (spread-spectrum watermark) encodes additional information within the original message without excessive extension of the payload. It is sufficient to extend the original message to accommodate the watermark. Without the watermark, we would have had to transmit the additional information as an additional message, which requires much more data to be sent.

Since confidentiality is one of the major concerns in any communication, researchers proposed many innovative solutions to improve it. Confidentiality through PLS was first considered in 1949 by Shannon, who presented the first application of information-theoretic secrecy [34]. On the other hand, cryptography, steganography, and watermarking are techniques that implement data secrecy using different paradigms than PLS. This paper uses watermarking as a

TABLE 1
Hypothesis in Long-Term Key Distribution Phase of the State of the Art Work on CAN Authentication Protocols

	CANAuth [35]	Car2X [36]	LCAP [4]	Libra-CAN [37]	CaCAN [38]	VeCure [39]	LeiA [10]	VatiCAN [40]	VulCAN [41]	Fassak et al. [42]	TOUCAN [9]	TACAN [17]	CANTO [19]	Palaniswamy et al. [43]	Xiao et al. [44]	Mun et al. [45]	CAN-TORO [46]	Youn et al. [47]	S2-CAN [48]	AutoSec [49]
Omit long-term key distribution?	✓		✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓
Omit long-term key refresh?	✓		✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓
Requires additional nodes?		✓		✓	✓			✓			✓		✓	✓						

covert-channel that is not part of the CAN protocol to transfer information between two devices. Such covert-channel is an essential part of the PLS solution that we propose here.

3 Related Work

The insecure-by-design nature of the CAN protocol attracted the attention of many security researchers. Several works in literature proposed different approaches to increase the security of the transmission over the CAN bus ranging from the re-design of the protocol to the implementation of IDSs [50]. However, many challenges should be considered when designing a CAN solution. First, the ECUs have very low computation power, hard real-time constraints, and typically 8 bytes of payload. Thus, solutions with heavy computational requirements (e.g., Asymmetric Encryption) are not well suited. Second, the intra-vehicular architecture may sometimes be difficult or even impossible to modify after the product deployment. Therefore, solutions requiring additional nodes may be unsuitable and impossible to implement. These factors make the design of a new CAN security solution challenging. In the following, we briefly summarize the different security solutions proposed in the literature, grouping them by the implemented features.

Intrusion Detection Systems. IDSs represent a cost-effective solution to monitor passively or actively the system's behavior. This solution requires a preliminary training period and is specific for the particular system implemented since similar systems may differ in time constraints. Furthermore, the IDS approach requires installing the detector on a node with a high computational capability and access to the entire bus communication. IDSs based on traffic analysis [11], [51]) or physical invariants such as clock skew [52], bit-based [53] or frame-based [12], signal characteristics [13], [54]. If on one hand, IDSs represent a flexible solution able to prevent zero-day exploits, on the other hand, IDSs may require a modification of the CAN architecture and notable computing power.

Security-by-Design. The redefinition of the entire protocol allows protecting communication most robustly. However, this solution requires a profound modification of the existing protocol and, consequently, an adaptation of all the current systems currently implementing it. Different researchers attempted to introduce cryptographic primitives, such as authentication or encryption. The most common approach is to implement a Message Authentication Code (MAC) in the CAN communication (e.g., [4], [10], [35], [38]). The two main approaches used in literature to send the MAC in the CAN environment are to send it as a separate message or to truncate it. The drawback of the first approach is that it requires an additional frame for every message, slowing the entire communication. Instead, the second approach considerably reduces the already short payload available of the CAN frame.

Covert-Channel Authentication. The main idea behind these techniques is to embed secret and unique information in the CAN frame, exploiting protocol properties to authenticate the sender node or the message. The main advantage of this class of techniques is that authentication information is embedded in the frame that carries the data without requiring an additional authentication frame and without increasing bus load. In [55] the authors present a security mechanism that exploits a covert-channel to implement a secure authentication between an ECU and the Monitor Node to generate shared session keys. Each ECU embeds unique authentication frames into CAN frames and continuously transmits them through covert-channels, which can be received and verified by an additional Monitor Node. Other works present covert-channel techniques to implement message authentication exploiting CAN transmission temporal features [18], [19], [20].

3.1 Limitation of Related Works

Although many algorithms introduce innovative authentication schemes for CAN networks, in most cases, these algorithms omit the implementation of a secure key exchange mechanism or assume that the key is installed in a secure storage partition at the vehicle production in *una tantum*. However, these assumptions are not flexible and do not consider the key refreshment during normal functioning in case of compromising. In Table 1 we report a summary of the assumptions on the key exchange of the various papers proposing authentication mechanisms over CAN bus (i.e., transmitter authentication or message authentication). We refer to *key exchange* as sharing the long-term secret that can be used, for instance, to validate the MAC. To the best of our knowledge, only in Car2X the authors consider a dynamic key exchange phase at the vehicle boot time. However, this solution requires adding a central node to manage the entire sharing process, with a consequent modification of the CAN protocol communication phases.

Contrarily, our solution aims at solving the lack of key exchange solutions by modifying the first phase of the communication at the vehicle boot time and without changing

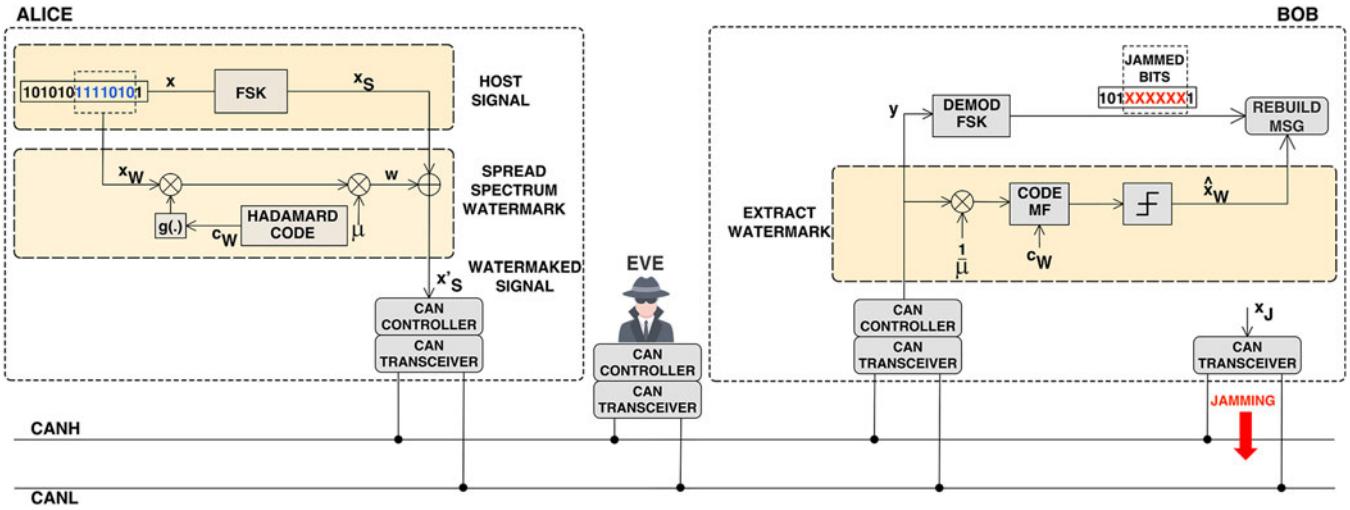


Fig. 3. WBPLSec system model on CAN, the transmitter (Alice), the receiver with jammer (Bob) and the adversary (Eve).

the existing architecture. Our methodology uses features already present in the CAN protocol and requires only modifying the source code in the ECUs.

Few works propose mechanisms to exchange the long-term secrets in a CAN network. We recall these works and compare them with SENECAN in Section 8.

4 System Model

In this section we present the system model for which SENECAN is conceived. In particular, in Section 4.1 we provide an overview of the key distribution mechanism based on jamming and watermarking. Instead, in Section 4.2 we describe the implementation of the WBPLSec algorithm in a real testbed to secure CAN communication. In general, the WBPLSec algorithm successfully applies to those media where jamming is possible. However, the effect is different between a wireless and a wired connection. In the former, the legitimate receiver can create a security region around him through jamming and watermarking [56]. In the latter case, the information is erased with jamming and then reconstructed with watermarking for all nodes connected to the bus. Furthermore, in wired communication, we no longer have the concept of a spatial region of security around the node that is jamming.

4.1 CAN Jamming Receiver

As previously described, CAN implements a CSMA/CR collision revolving mechanism based on ID priority. When a collision occurs, the dominant bit "0" wins over the recessive bit "1". What happens at the physical level is that, when transmitting a bit "0", CANH and CANL are set respectively at $3.5V$ and $1.5V$, with a resulting $\Delta V = 2V$. Instead, when transmitting a bit "1", CANH and CANL are set respectively at $2.5V$ and $2.5V$, with a resulting $\Delta V = 0V$. This means that if both "0" and "1" are transmitted simultaneously, the bus will have a final $\Delta V = 2V$ and, therefore, all the nodes will see a "0". A node can transmit a series of "0" on purpose during another transmission to perform jamming by destroying a part of a frame.

Based on this principle, we develop an authenticated key distribution mechanism that exploits the jamming principles at the physical layer over wired communication to destroy part of the frame selectively. In particular, when Bob notes that Alice is trying to send him a frame, Bob starts jamming the communication so that only he can reconstruct the destroyed original message. To reconstruct the partially destroyed message, we implement the WBPLSec algorithm, which will be described in Section 4.2. Since this mechanism expands the size of the message, introducing an overhead on the transmission, it can be used to exchange a secret between the nodes (e.g., when the system starts) to encrypt future messages with traditional symmetric encryption algorithms (e.g., MAC and AES), introducing security properties in the communications.

4.2 WBPLSec Algorithm Applied to CAN

Since small sensors have limited computation power, in 2017, Soderi *et al.* developed the WBPLSec protocol for wireless communications as a valuable physical layer security standalone solution [22]. This technique combines watermarking as a covert-channel and a jamming receiver.

CAN is a robust vehicle bus for networking intelligent devices. Like the wireless sensors, the nodes of a CAN network have limited resources. The main intuition that inspired this work was to provide a new security solution for CAN network nodes fully compatible with the CAN standard stack. Indeed, in this paper, we investigate the application of WBPLSec for the first time on a wired bus.

The scheme in Fig. 3 depicts the proposed system model for the CAN protocol and is based on four main actions to transmit messages securely through the CAN: (1) spread-spectrum watermarking: part of the secret message is first modulated with a spreading sequence and then summed with the host signal; (2) receive jamming: Bob disrupts only part of Alice's message using an additional CAN transceiver; (3) message reconstruction: Bob knows the jammed part, he can reconstruct the clean message. Jamming does not affect the spread-spectrum watermark; (4) watermark extraction: we extract the watermark using a matched code

filter (CMF) that uses the same spreading code (i.e., c_W) used in transmission.

With reference to Fig. 3, let's assume that the transmitter, i.e., Alice, and the receiver, i.e., Bob, would exchange a *secret message*, i.e., x, for supporting future confidential transmissions. WBPLSec transmits the information via two independent paths implementing a data splitting policy. Thus, the information is sent via a narrow-band signal and through the SS watermarked signal. Without loss in generality, in the rest of the paper, we use the Direct Sequence Spread Spectrum (DSSS) for watermarking and a Frequency Shift Keying (FSK) as narrow-band signal.[1]

Bob partially jams the watermarked signal, but this interference only affects the narrow-band signal because the SS watermark is immune to this type of interference. In this way, the watermark is used to recompose the entire signal [22].

The ECU transmitter combines the original modulated signal, x_S, with an SS watermark, w. In particular, we select the first equation for watermarking defined by Cox *et al.* [33] to build the watermarked signal as follows

$$x'_S(i) = x_S(i) + \mu w(i), \tag{1}$$

where $x_S(i)$ is the ith bit of the continuous FSK transmitted signal [57], μ is the scaling parameter and $w(i)$ is the SS watermark. The signal watermarking is generated by using the traditional spread spectrum-based approach [58].

The host FSK modulated signal x_S can be expressed as

$$x_S(i) = A_a \sqrt{\frac{2}{T_{hs}}} \cdot cos(2\pi f_n i), \quad \text{for } 0 \leq i \leq T_{hs}, \tag{2}$$

where A_a is the amplitude and T_{hs} is the symbol time. Then, $f_n = f_c + \frac{1-2b}{2T_{hs}}$ indicates the two frequencies needed to transmit two ($n = 1, 2$) binary digits (b). We assume equal to 0 the initial phase of FSK signal.

Recall that Alice and Bob want to exchange a secret x. This secret is modulated FSK to create the host signal while a part, x_W, is used to create the SS watermark. We select the last N_W over N bits from x to create x_W, which is given by

$$x_W(i) = \begin{cases} x(i), & \text{for } N - N_W \leq i \leq N, \\ 0, & \text{elsewhere.} \end{cases} \tag{3}$$

The DSSS watermark signal can be expressed as

$$w(i) = \sum_{k=-\infty}^{+\infty} \sum_{j=0}^{N_c-1} g(i - kT_b - jT_c)(c_W(i))_j (x_W(i))_k, \tag{4}$$

where $(x_W(i))_k$ is the kth bit of the watermark signal. $(c_W(i))_j$ represents the jth chip of the orthogonal Pseudo-Noise (PN) sequence. $g(i)$ is the pulse waveform, T_c is the chip time, and $T_b = N_c T_c$ is the bit time.

Next, the watermark embedding step in the host FSK signal, Equation (1), occurs with w signal modulating a carrier

1. We can obtain similar results if we select Amplitude Shift Key (ASK) for the host signal and DSSS for the SS watermarking.

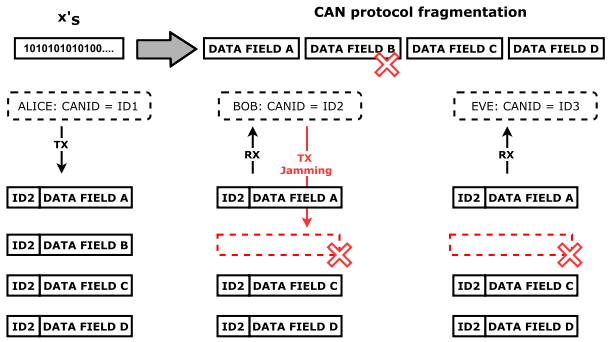


Fig. 4. WBPLSec jamming operational diagram on CAN.

frequency close to the range of the f_c used by FSK. The watermarking operation is performed by the ECU sender (Alice) before the transmission over the bus. Fig. 3 shows how the transmitter embeds the original message into the host signal. Then, the processor passes the watermarked signal to the CAN controller that splits it into CAN frames. According to the dimension of the frame allowed by the CAN protocol, each frame's data field contains up to 64 bits.

As shown in Fig. 4, while Bob is receiving CAN frames with his arbitration ID, he can jam at most $M = N_W$ bits because these N_W bits are transmitted through SS watermark, i.e., through the covert-channel. *Jamming a signal in the CAN protocol* is equivalent to transmitting a dominant bit on the bus, i.e., the bit "0". For simplicity in our experiments, we destroy the entire frame, and therefore we destroy blocks of 64 bits of the data field. The effect of the jamming will be to destroy the whole frame (i.e., all the fields), canceling it from the bus for all the receivers, including the attacker (Eve), connected on the same bus. Note that the distance between Eve and the other nodes does not influence the attack scenario since the propagation speed of the information can be assumed as the velocity of light. The receiver has two *fingers* as shown in Fig. 3. Bob demodulates the FSK signal with the first, whereas he recovers the SS watermark with the other. Unfortunately, part of the demodulated information is corrupted due to the jamming. However, Bob can get a clean signal by replacing corrupted bits with the information he conveys via the SS watermark that is immune to any jamming interference. In contrast, the eavesdropper cannot remove the interference because he does not know the jamming characteristics and which packets frames are affected.

The physical layer security procedure, WBPLSec, used by two ECUs to exchange the secret, is depicted in Algorithm 1. Note that we assume perfect synchronization between Alice and Bob, so that the two ECUs know when the secret is sent. The synchronization between the nodes will be discussed in Section 5.1. In addition, we know that the legitimate receiver should not jam more bits than those Alice embeds into the watermark, i.e., $M \leq N_W$. Indeed, the WBPLSec algorithm replicates a portion of the original information through the watermark, i.e., N_W of the total N bits. In this way, N_W will also be the maximum number of usable bits that we can use to reconstruct the information destroyed through jamming, i.e., M bits. So, in summary, to not lose information due to jamming, we must have $M \leq N_W$.

Algorithm 1. WBPLSec Algorithm in CAN Protocol

1: **procedure** KEY DISTRIBUTION THROUGH PLS
2: **Input**: x_S
3: **SS Watermarking (ALICE)**:
The original message is modulated FSK.
One part of the original message is first modulated with DSSS and then embedded into the host FSK signal.
4: **Transmission (ALICE)**:
The FSK watermarked message is sent to the Alice's CAN controller and then on the bus.
5: **Jamming Receiver (BOB)**:
The receiver jams M bits transmitted by Alice through a second CAN Transceiver.
The received signal is then processed by the FSK demodulator to recover the data transmitted through the CAN.
Due to the jamming, part of the received signal is now corrupted and unusable.
6: **Watermark Extraction (BOB)**:
The receiver extracts the watermark from the received signal by using a code matched filter.
7: **Symbol Rebuild (BOB)**:
Knowing which bits are jammed the receiver, i.e., Bob, is able to rebuild a clean symbol using information contained in the watermark.
8: **Output**: x'_S
9: **end procedure**

5 PROTOCOL IMPLEMENTATION

In the following, we propose the first implementation of WBPLSec over the CAN bus in Section 5.1 discussing the implementation choices to prove the reliability of this algorithm in a wired bus. Then, in Section 5.2, we present an analysis of the performance of this algorithm in a virtual environment.

5.1 Functioning

The key distribution phase is performed entirely in a dedicated time slot. The total length of such slot corresponds to the total number of single key distribution sessions, i.e., the unordered number of couple of nodes communicating with each other (i.e., if Alice sends messages to Bob and vice versa, this is a distribution session since the shared key is symmetric) multiplied by the time of a single key distribution. This requirement will be discussed in Section 8. In this phase, every node has a set of defined time slots to send the keys to all the other nodes that are supposed to communicate within successive phases. This dedicated time slot is necessary to obtain a communication order and synchronization in the key exchange to avoid multiple simultaneous jamming from more nodes. The organization of this phase depends on the specific number of nodes composing the network and their communication scheme. Therefore, this phase can be configured by the system manufacturers. We assume that each ECU stores a list to associate every other ECU to a particular KEY-ID number. Such a list is configured by the network manufacturers and stored in a secure and tamper-resistant area of the memory. The reason behind introducing the KEY-ID is because the standard

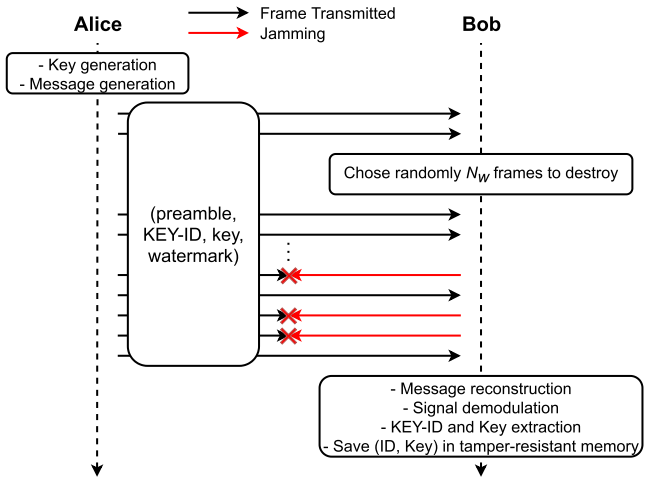


Fig. 5. Protocol steps representation.

CAN frame does not allow to specify the sender node. Thus, the receiving node will not be able to associate the corresponding decryption key. By including the KEY-ID in the message sent, every receiving ECU knows who is currently transmitting. Consequently, the receiving ECUs can associate the received symmetric key with the sender KEY-ID and use it for future communication. Future communication must reserve part of the data field (i.e., 8 bits) to include KEY-ID to authenticate the sender. Furthermore, since the CAN broadcast communication is based on an arbitration ID, each ECU with the same arbitration ID should also share the same key for a specific KEY-ID. More precisely, if a node sends a message to a specific arbitration ID, all nodes with such arbitration ID will receive the message. Therefore, all the receiving nodes with the same arbitration ID must know the same key to decrypt the message.

At high-level, the steps required to perform the communication are reported in Fig. 5 and described as follows.

1) Alice wants to securely send a key to Bob in the predefined time slot. The time slot length is fixed and corresponds exactly to the estimated time to complete the key transmission, i.e., complete the steps below.
2) Alice FSK modulates the original key and adds the signal's watermark. Furthermore, to prevent the retransmission due to the frames collision, she disables the default transmission queue capabilities (i.e., if the bus is busy, the ECU will not re-transmit the frame later). The KEY-ID of Alice is inserted into the original message to authenticate the nodes correctly. We decide to reserve the last 8 bits of the preamble to the KEY-ID of the transmitting node to support up to 256 different nodes on the bus. However, this parameter can be easily extended.
3) While Alice is transmitting frames, Bob, who knows by design that he is the receiver in this time slot, starts jamming using the additional transceiver to destroy a random number of frames. Since Alice is a node with no transmission queue, all the jammed frames will be lost and not re-transmitted once jamming is finished. While jamming, Bob stores the received frames in a queue.

TABLE 2
Parameters for WBPLSec Experiments Over CAN

Parameter	Value
FSK frequencies[1]	9.75 kHz, 10.25 kHz
DSSS carrier frequency	12 kHz
Samples[2] per FSK symbol (sl)	80
Samples[2] per DSSS symbol (sl)	80
Number of bits FSK payload (N)	128, 384
Number of bits FSK preamble[3]	128
Max. Number of jammed bits (M)	$2 \div 24$
Number of bits to create the watermark (N_W)	8, 24
Number of bits watermark preamble	8
Watermarking scaling parameter (μ)	0.3
DSSS Processing Gain (G_p)[4]	4, 8, 16

1 *Up-converted using 10 kHz carrier frequency.*
2 *We assume the same symbol length for FSK and DSSS signals.*
3 *It consists of the preamble and a synchronization sequence.*
4 *Using Hadamard PN code.*

4) Bob lost frames during step 3), but he is the only one with the knowledge of the jammed bits position and consequently able to recover the key transmitted by Alice. By merging the frames in the queue in order, Bob can reconstruct the original message with the KEY-ID and the key thanks to an FSK demodulator and a watermark extraction.
5) Lastly, Bob extracts and saves the couple (KEY-ID, key) received from Alice in a tamper-resistant memory.

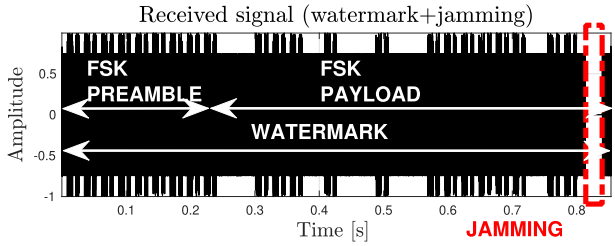
This procedure can be performed in the first phase of the network creation (e.g., vehicle pre-sale) or successive sessions to *refresh the keys* (e.g., during the vehicle revision or OTA updates), and it is repeated for every ordered couple of nodes. Again, this requirement will be discussed in Section 8. The procedure discussed allows the node to send or refresh the key securely. Furthermore, the schema enables distributing the keys in a one-way transmission way without requiring a response by the receiving node.

5.2 Simulation Performance

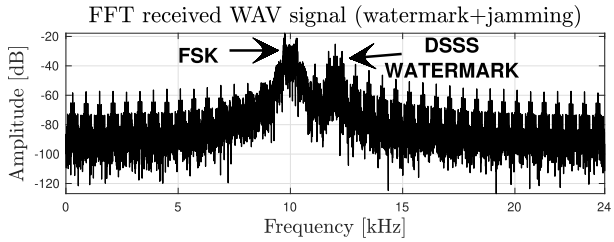
In this section, we evaluate the performance of WBPLSec over CAN bus in terms of Bit-Error-Rate (BER) of the watermarked signal x'_S and the watermark w. We simulate in Matlab 2021a the BITS architecture using the parameters in Table 2 to generate the signal with watermark and the relative jamming during transmission. We simulate the algorithm's key distribution behavior, including the jamming interference that produces the cancellation of CAN frames.

Considering the jamming on the bus, the simulations we perform are very similar to a real-world case. As described in the next section, the effect of jamming is to erase the frames on the bus, and therefore we can easily simulate it in a virtual environment. In this scenario, we can evaluate the effect of jamming on multiple transmissions by varying the length of the original message (N), the SS watermark (N_W and G_p), and the number of jammed bits (M).

The overall intuition of the BITS architecture is represented in Fig. 6. There, we show the watermarked signal (x'_S) in the Time Domain (TD) (Fig. 6a) and in the Frequency Domain (FD) (Fig. 6b). The TD white arrows denote the preamble and the payload, whereas the watermark is spread over the entire signal. Also, in red, we indicate the information canceled by the jamming interference produced by the legitimate receiver.



(a) Watermarked received signal with jamming in the TD ($M = 16$).



(b) Watermarked received signal with jamming in the FD ($M = 16$).

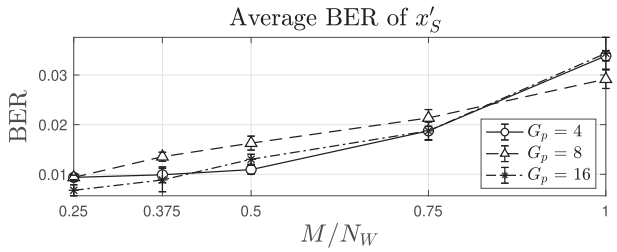
Fig. 6. BITS architecture over CAN.

In a critical view of our contribution, we should note that with this number of samples per bit, we get up to 40960 samples when x'_S is 512 bits long. In fact, we compute x'_S with (1), where we represent each bit with 80 samples which include the oversampling rate needed for simulations. By default, Matlab stores all numeric values in double-precision floating-point, representing each sample with 64 bits. Nevertheless, 64 bits is also the length of the CAN protocol data field. Therefore, we can transmit one sample at a time per CAN frame. This certainly increases the transmission. To give an idea of what could be a real-world transmission time with this schema, we report Table 4 with the theoretical transmission times calculated with the nominal speeds supported by the CAN standard.

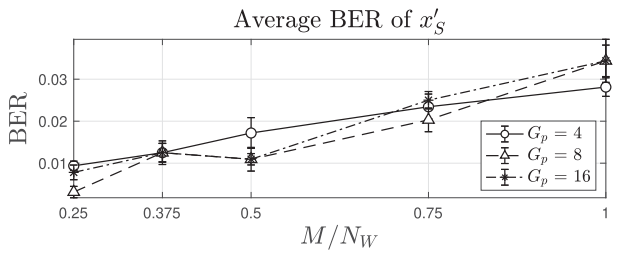
We verify the transmission quality through the WBPLSec algorithm on CAN in terms of BER. Fig. 7 shows the average BER and its accuracy (i.e., the standard error of the average) of the watermarked signal x'_S for two different message lengths and different watermark settings. In particular, we consider for both scenarios the same ratio $\frac{M}{N_W}$. Under these settings, we can see that the more bits are destroyed, the greater the BER of the FSK signal. And this is the effect intended by Bob, i.e., disrupting the communication with his jammer so that the attacker cannot receive the information sent on the channel by Alice.

Before evaluating the number of errors in the watermark after jamming, we extract it through a Code Matched Filter (CMF). This process is performed by computing the normalized statistics [22], [58]

$$r \triangleq \frac{\langle \boldsymbol{y}, \boldsymbol{c_W} \rangle}{\langle \boldsymbol{c_W}, \boldsymbol{c_W} \rangle}, \tag{5}$$

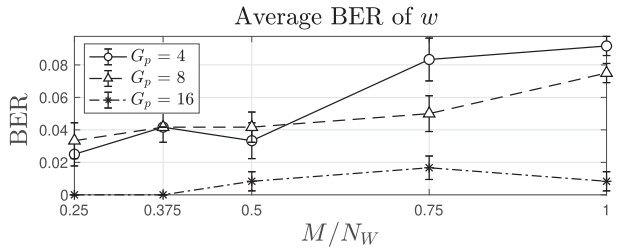


(a) $N = 384$ bits, $N_W = 24$ bits

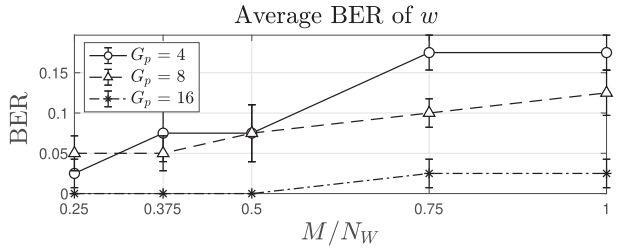


(b) $N = 128$ bits, $N_W = 8$ bits

Fig. 7. Watermarked signal (x'_S) average BER when jamming, i.e., M and G_p change.



(a) $N = 384$ bits, $N_W = 24$ bits



(b) $N = 128$ bits, $N_W = 8$ bits

Fig. 9. Watermark (w) average BER when jamming, i.e., M, and G_p change.

where the $\boldsymbol{y}$ is the received signal by Bob, as shown in the system model represented in Fig. 3, and $\boldsymbol{c_W}$ represents the Hadamard PN sequence. We assume $\langle \boldsymbol{c_W}, \boldsymbol{c_W} \rangle = 1$, i.e., PN sequences have unit energy. Fig. 8 shows the output of the CMF, where we can observe the eight preamble bits used to synchronize with the start of the watermark. It is known how the matched filter maximizes the ratio at the output of the detector. And, the detector is the same one that was introduced with the traditional SS watermarking [22], [58], and the estimation of the embedded bit is given by $\hat{x}_W = sign(r)$.

Finally, we observe that the watermark also undergoes interference. Fig. 9 shows the average BER and its accuracy (i.e., the standard error of the average) of the watermark w. Nevertheless, in Fig. 9, we can see how the information conveyed through the watermark is more immune to jamming. This is due to an inherent feature of spread-spectrum technology that is immune to narrowband interference, such as Bob's jamming. In particular, when the processing gain, or G_p, is at least equal to 16, the BER of w is less than 3%. This BER result is absolutely in line as it is obtained in other implementations of WBPLSec [24].

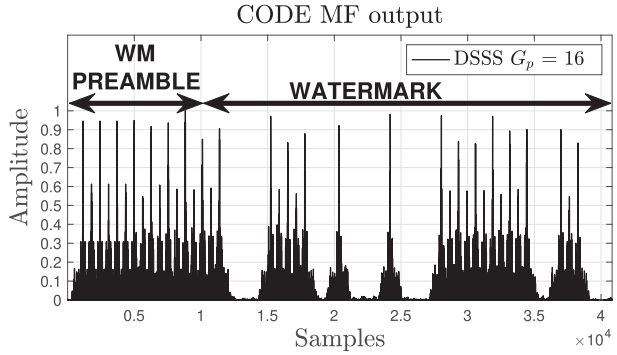


Fig. 8. Extraction of the watermark using a code matched filter to the spreading code ($N = 384$, $N_W = 24$, $M = 16$).

6 Testbed Validation

To test and validate the SENECAN functioning, we implement a scaled-down architecture composed of three nodes. In this section we describe the experiments we perform to reproduce the SENCAN distribution phase in a simplified real-world scenario. In particular, in Section 6.1 we describe our testing environment and the devices used. As an example scenario, to perform the tests we set $N = 512$ bits (i.e., 40960 samples in double precision, hence, 40960 frames), $G_p = 16$ and we jam 16 consecutive bits ($M = 16$), which correspond to jam $16 \cdot 80$ frames.

6.1 Equipment Setup

The architecture used to perform the experiments, represented in Fig. 10, is composed of three nodes: Alice, Bob, and Eve. Alice and Bob consist of an Arduino UNO board and a CAN shield from SeedStudio, respectively. Each CAN shield consists of a Microchip MCP2515 CAN controller and an MCP2551 CAN transceiver. Furthermore, Bob contains an additional MCP2551 CAN transceiver that we use to perform the jamming interference, as the implementation by Palanca *et al.* [59]. Instead, Eve consists of a

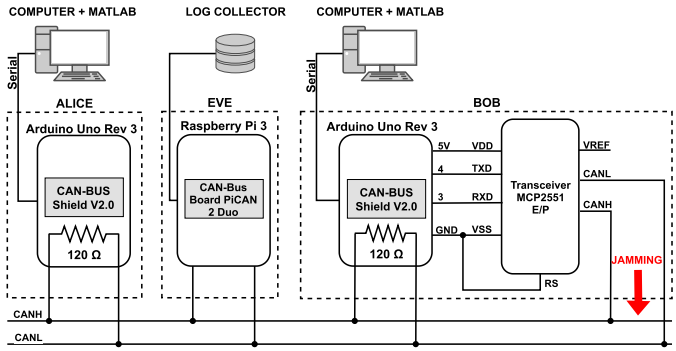


Fig. 10. Experimental setup used for the proof of concept with the hardware selected for the testbed.

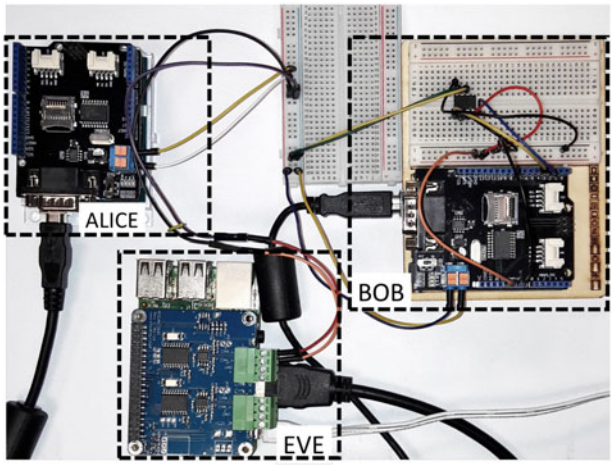


Fig. 11. Real experimental setup with three nodes.

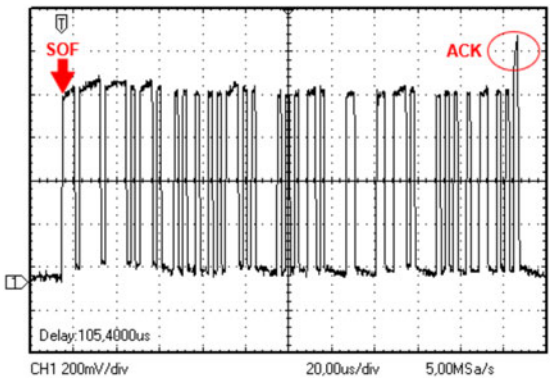


Fig. 12. Representation of a frame measuring CANH and CANL under normal conditions. The image is collected from the oscilloscope and a differential probe. The first bit is the start-of-frame that denotes the start of the frame.

Raspberry Pi 3 and a PiCAN2 DUO board that provides a CAN bus interface. The decision of using a Raspberry Pi to represent Eve is motivated by the necessity of easily collecting the traffic passing through the bus with a third node (i.e., Eve). To do this, we install Wireshark on Eve. Wireshark is a network protocol analyzer able to inspect and analyze CAN bus frames, and it gives us the possibility to monitor and collect traffic over the bus.

To perform the experiments, we set the transmission rate at 500 kbps, the common bitrate of high-speed CAN. Whereas the high-speed CAN (i.e., ISO 11898-2 [27]) network requires only two 120 Ω terminal resistors, we use the resistors embedded in the Arduino CAN shield.

Algorithm 2. Jamming Loop in Bob

Input: $N; N_W; s_l$
Output: Deletion of $N_W \cdot s_l$ CAN frames.
$jam_f :=$ Random(1, $N_W \cdot s_l$); ▷Frames jammed
$jam_s := (N - N_W \cdot s_l - jam_f)$; ▷Start of jamming
$c := 0$; $frame_{rx} := 0$;
while $c < jam_f$ **do**
  if $frame_{rx} \leq jam_s$ **then**
    $frame_{rx} = frame_{rx} + 1$; ▷No Jamming
  else
    WriteBit(0, Port, Pin); ▷Write Dominant Bit
    Wait(149 μ);
    WriteBit(1, Port, Pin); ▷Write Recessive Bit
    $frame_{rx} = frame_{rx} + 1$;
    $c = c + 1$;
  end
end

6.2 Implementation

The implementation of the experimental architecture is depicted in Fig. 11. In our setup, Alice communicates with a base frame format over the bus and uses a fixed identifier for all the frames. To perform the *jamming*, Bob occupies the channel with the dominant bit to prevent communication to all the other nodes in the network. However, the default driver of the CAN transceiver MCP2551 prevents holding a dominant bit for more than 1.25 ms. In this case, the controller called TXD Permanent Dominant Detection inside the MCP2551 transceiver will disable the CANH and CANL output drivers to prevent data corruption on the CAN bus. We decide to address this issue by introducing recessive bits during the jamming. The CAN frame implements the practice of bit stuffing. An example of the experimental setup frame is shown in Fig. 12. The overshoot at the end of each frame depends on the acknowledgment (ACK) bit, which is dominant, and all the nodes drive it except for the one transmitting the frame. Because more nodes drive the bus dominant, a higher voltage is observed on the bus at the end of each frame. The communication of Alice is handled in order not to create a frame queue. In particular, when Alice cannot transmit the frame (e.g., jamming in progress), she will refuse to transmit the next frame. Moreover, the data field of each frame is transmitted to Alice via serial communication from Matlab. Implementing serial communication with Matlab arises from the need to easily store a significant number of samples and facilitate data processing. Each double value of the bitstream stored in Matlab is communicated to Alice in a little-endian representation, and 8 ms delay is added between the serial writing of this value. For a correct reception of the bytes from serial, Alice will be necessary wait for 1 ms for each byte before forwarding the frame on the CAN network. Therefore, due to the delay introduced by the implementation setup in Matlab, frames are sent on the bus every 15 ms. Bob is also connected independently to a device with Matlab to store and process the values received from Alice. This time, the serial communication is directed from the Arduino board to Matlab. The Arduino board related to Bob can read the CAN frames through the dedicated CAN shield, count the frames to activate the jamming via the independent MCP2551 transceiver, and print the received frames on the serial line. Once Bob receives the predetermined frame before jamming, he starts to occupy the channel with the dominant bit for the entire time window necessary to destroy the predefined number of frames as sketched in Algorithm 2. As mentioned, Bob cannot write a dominant bit for more than 1.25 ms, but frames arrive every 15 ms on average. For this reason, Bob alternates recessive bits to cover the time window of a single frame. In our setup, we decide to keep the

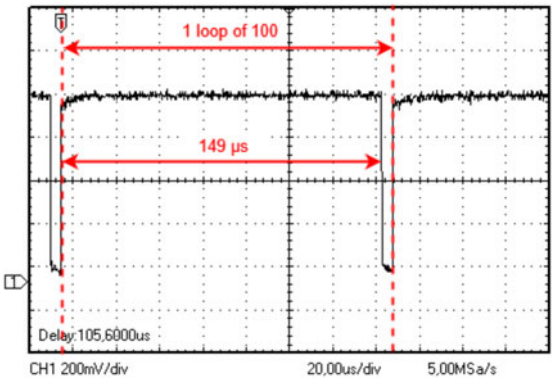


Fig. 13. Representation of the CANH and CANL under jamming condition. The image is collected from the oscilloscope used in our experiments and a differential probe.

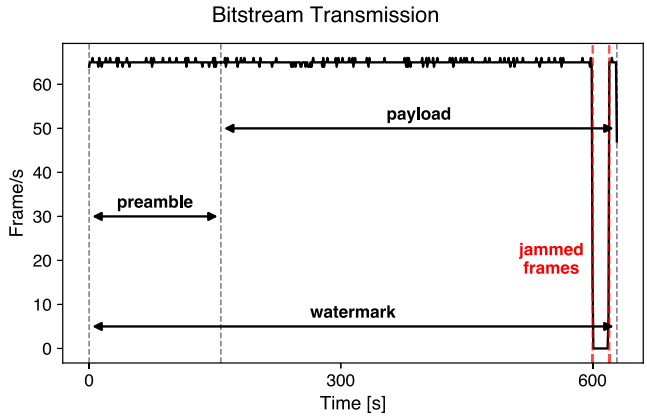


Fig. 14. Bitstream transmission in terms of frame per second. The Figure shows the different fields of the bitstream and the jamming window ($N = 512$, $N_W = 24$, $M = 16$, $G_p = 16$).

dominant bit for 149 μs a hundred times to destroy a single frame, as shown in Fig. 13. The Fig. 14 shows the effect of jamming on the bitstream transmission.

7 Secuirty Analysis

In this section we analyze the security of the SENECAN key distribution mechanism. In Section 7.1 we discuss the possible threat model and attacker capabilities based on the most common assumption in the literature. Then, in Section 7.2 we analyze the security properties which our methodology ensures.

7.1 Threat Model

To evaluate the robustness of our approach, we assume that an attacker can access the communication over the CAN bus and perform different actions at every moment of the key distribution process. In particular, we consider the scenario where an attacker physically accesses the network during the distribution process via the bus. In this setting, the attacker can both passively sniff or modify the communication. As previously described, we assume that a list to associate every ECU with a corresponding KEY-ID is stored in every ECU's secure and tamper-resistant memory. This assumption is consistent with the other works in this field, where the long-term secret is assumed to be securely stored during the production phase. Unlike the other works, we assume that the pre-shared secret consists only of a list of KEY-IDs used to associate the transmitting ECU with the corresponding KEY-ID. This corresponds to a *weak secret*. According to the literature, the common attacks that an attacker can perform in a CAN communication are the following.

- *Message Injection Attack*: The goal of this attack is to send a customized and malicious frame to an ECU to compromise the authentication phase.
- *Replay Attack*: This attack aims to reuse a previously transmitted and sniffed frame in successive communication to replicate the legitimate transmission.
- *Message Modification*: In this case, the goal is to modify the frame during transmission. This can be done in real-time by selectively modifying jamming the communication to flip some bit from 1 to 0, or by collecting the entire message, modifying it, and retransmit it.
- *Eavesdropping*: This attack aims at passively sniffing the communication to collect the traffic during the key exchange and, for example, to analyze it in a second moment to compromise future communication.
- *Masquerade Attack*: The attacker can modify and reuse the transmitted KEY-ID to impersonate an ECU during the exchange phase.
- *Adversarial Jamming*: An attacker can leverage a disturbing interference (e.g., jamming) similar to our mechanism to disrupt the communication, destroying one or more frames. In this way, an attacker can obtain a denial of service by removing part of the exchanged message.

7.2 Security Properties

In the following, we discuss the security features of SENECAN. For each property, we then state which of the attacks described in the previous section can prevent. As previously mentioned, our solution can exchange a long-term shared secret or refresh it. This procedure ensures the following security properties.

Previous Secret Independence. If the key currently used is compromised, SENECAN allows to refresh the keys independently by the previous keys. Therefore the information obtained by an attacker could not affect the new secret exchange. The only requirement of the procedure is that the association of the KEY-ID with each other node is not compromised. This property allows preventing *Message Injection Attacks*.

Replay Attack Resistance. It is probably the most challenging feature required for an offline authentication schema. Generally, this feature requires a nonce exchange and synchronization by the different nodes. This is a very complex constraint to obtain, as explained in [60]. Different works address this problem with an additional node and a constant synchronization protocol. In our case, the replay protection is ensured because Bob chose and destroyed a random frame set. This information is known only by him, and the next time a new secret is shared, the jammed frames will be different. This will avoid any attempts by Eve to reuse an old frame. This property allows to prevent *Replay* and *Message Injection Attacks*.

TABLE 3
Different Key Distribution Approaches for CAN

	Car2X [36]	Mueller *et al.* [21]	Jain *et al.* [61]	TRICKS [20]	SENECAN
Previous Secret Independence		✓	✓	✓	✓
Replay Attack Resistance		✓	✓	✓	✓
Confidentiality	✓	✓	✓	✓	✓
Integrity			✓		✓
Authentication	✓		✓	✓	✓
One-Way Transmission		✓	✓		✓
No Additional Node		✓		✓	✓
Implementation	✓			✓	✓

High Confidentiality. The confidentiality of the communication is achieved thanks to the jamming phase, which allows only Bob to know the jamming points and consequently to reconstruct the original message. Since every bit of the original message is expanded to 80 frames, 64 bits payload each, to brute force the original message, the attacker requires to compute $2^{M \cdot 80 \cdot 64}$, where M is the number of the bit jammed during the frame transmission. The security level of the schema is consequently $M \cdot 80 \cdot 64$. This property allows preventing *Eavesdropping*.

Integrity. there are two possibilities for an attacker to modify the original message: in real-time or in a secondary moment. While the first approach is unfeasible due to the anti-replay property, the second would compromise the watermark, raising errors during the demodulation and watermarking verification. This property prevents *Replay*, *Message Injection Attacks*, and *Message Modification Attacks*. Furthermore, it can also identify *Adversarial Jamming* attempts.

Authentication. We assume that the list of the KEY-IDs of each node is stored in a secure memory area of the ECU and therefore is not possible to modify. By adding this trusted and confidential information in the message exchanged, we ensure the authentication of the transmitter and to store in the receiver the corresponding couple KEY-ID and Key. The KEY-ID could also be used in future MAC mechanisms built on top of this solution. This property allows to prevent *Masquerade* and *Message Injection Attacks*.

*Malicious Interference Resistance.*Thanks to the information in the watermark, SENECAN can partially mitigate *Adversarial Jamming* attacks. In fact, in the most favorable case, if the adversary would destroy the frames $N_W \cdot 80$ frames dedicated to transmit the watermark, Bob can still reconstruct the original message using the Algorithm 2. On the contrary, if Eve jams other frames, SENECAN will still suffer from this type of attack. A manufacturer can increase the number of bits N_W for the watermarking process, increasing the attackable frame number for the attacker, in exchange of an overhead due to more information to transmit.

8 Discussion

As described in the previous section, SENECAN allows obtaining all the security properties. Furthermore, there is no need to add nodes that act as a trusted third part differently from many previous works. Also, our approach implements a one-way communication rather than bidirectional communication (i.e., challenge-response mechanism) differently from other works. This is a valuable property, especially in a broadcast environment as CAN, where the sender identity is not declared in the message. Our paper proposes a schema to securely exchange or refresh the long-term shared key. Due to the high computation requirements, we propose to perform this operation in a non-critical dedicated session (e.g., after the car turns on, in an OTA update, or during car maintenance). However, there can be other reasons to modify the long-term keys. For example, a manufacturer may require refreshing all the nodes' keys to increase their length or change the cryptographic primitives. The manufacturer can also refresh the keys in case of compromise. After the master secret is shared or refreshed, a car vendor can implement other security schemes on top of SENECAN for the successive phases of the communication. Therefore the specific key lifespan can depend on the security policies in the particular application.

Comparison With Other Schemes. As discussed in Section 3.1, most of the works assume that the long-term secrets are shared in the ECU during the production phase and do not consider the refresh of such secret in a successive phase. The only possibility to modify such a secret is physically replacing the ECU. To the best of our knowledge, few works proposed specific approaches to exchange and refresh long-term secrets in the CAN network. In Table 3 we report the comparison between the other works in literature that propose CAN-specific key distribution mechanism and SENECAN. The comparison reports the security features discussed in Section 7.2. It is important to note that not all the works discussed the mentioned properties. Therefore the analysis is based on our interpretation of the different works. We include in the comparison the type of communication (i.e., one-way transmission or not) and if the approaches require an additional node acting as a supervisor or a trusted third party for the distribution process. This last property is essential since it does not modify the network architecture and insert additional nodes. We also compare if the approach proposed was implemented and tested in a real environment or only discussed theoretically. In fact, in [21], [61] the authors proposed a collision-based approach similar to ours. However, their schema does not include integrity and authentication features but was not implemented and validated in a real scenario. Therefore, the requirement relative to the additional transceiver was not considered and discussed. We must note that even if SENECAN implements all the security properties mentioned and overcome their constraints, all the other approaches require fewer frames to transmit during the distribution, and consequently, they are much faster.

TABLE 4
Transmission time comparison

Bitstream	Samples	Transmission Time	Time[1]	Optimised Time[2]
256 bits	20480[3]	7200 bps[4]	307.2 s	76.8 s
		500 kbps	4.42 s	2.21 s
		1 Mbps	2.21 s	1.11 s
		10 Mbps	0.22 s	0.11 s
512 bits	40960[3]	7200 bps[4]	614.4 s	153.6 s
		500 kbps	8.85 s	4.42 s
		1 Mbps	4.42 s	2.21 s
		10 Mbps	0.44 s	0.22 s

1*One sample in double precision for each CAN frame.*
2*Two samples in single precision for each CAN frame.*
3*80 samples for each symbol.*
4*Proof of concept with 15 ms delay for each frame.*

Limitations. We decide to transmit data through the serial to manage the bitstream with Matlab in the testing phase for implementation simplicity. However, this introduce a bottleneck in the exchange data between Matlab and Arduino, making each frame delivered every 15 ms, equivalent to an estimated 7200 bps. This time is also due to the dominant bit holding limitation. This overhead is even more impacting if considering the total number of key distribution sessions. By considering the KEY-ID field of 8 bits, in the limit case, we support up to $N_{ECU} = 256$ ECUs. Since every pair of ECUs must share a common key, there will be $(N_{ECU} \cdot (N_{ECU} - 1))/2 = 32640$ key distribution sessions. Considering the setting used in our experiments and reported in Table 4 if the bitstream is 256 bits, this would require $32640 * 307.2\ s \simeq 2785\ h$, which is clearly unfeasible. In particular, Table 4 reports the transmission time with our scenario and the estimated time with the typical transmission speed used in CAN environments. We believe that with a dedicated hardware implementation, i.e., the computation is entirely implemented in the ECU, without the necessity of the serial communication and the possibility to hold the dominant bit for the desired time, the transmission time can be reduced as shown in Table 4. In particular, with dedicated hardware, we can avoid the 115.2 kbps bottleneck due to the serial communication and the need to wait for 15 ms for every frame due to the Arduino driver. Furthermore, since the double-precision representation used by Matlab overestimates the representation of the signal, to reduce the transmission load, every sample could be converted into a single-precision representation. This improvement allows transmitting two samples in a single frame, thus reducing by half the overall number of frames to transmit. With these simple optimization, we can cut the communication time as reported in Table 4. Considering a common CAN transmission speed of 500 kbps, the limit case transmission time would be $32640 * 2.21\ s \simeq 20\ h$. However, we believe that this number is still largely overestimated since not all the ECUs need to communicate with each other. Finally, our solution requires every ECU that implements SENECAN to embed an additional transceiver. This is because an ECU requires an interface to perform the jamming and another to read the bus simultaneously.

9 Conclusion

In this paper, we present and prove the effectiveness of SENECAN, an authenticated key distribution framework for devices communicating through the CAN bus network. The presented mechanism exploits the default protocol's bit collision properties to selectively destroy the frames by jamming over a wired connection. The experimental setup we use represents a proof of concept for future work in this direction and can be employed by all the devices operating in the CAN network to exchange a long-term secret among the different nodes. Moreover, in exchange for a long distribution phase, which can be executed in a defined phase of the system's life-cycle (e.g., after the production or during the vehicle revision), our approach allows obtaining more robust security properties compared to similar works and with a minimal network modification. Despite the reduced transmission speed in the SENECAN proof-of-concept implemented, we validate the key distribution based on the WBPLsec algorithm. As previously discussed, we strongly believe that the limitations introduced by Matlab and Arduino can be overcome by introducing ad hoc hardware and optimization. However, we do not exclude that the presence of dedicated hardware could guarantee Bob to jam with a single transceiver assigned to the reception of frames in a normal traffic scenario. SENECAN is therefore helpful for systems with a limited number of devices communicating with and can be used as a building block for future CAN-based secure systems.

References

[1] K. Iehira, H. Inoue, and K. Ishida, "Spoofing attack using bus-off attacks against a specific ECU of the can bus," in *Proc. 15th IEEE Annu. Consum. Commun. Netw. Conf.*, 2018, pp. 1–4.

[2] K. Koscher *et al.*, "Experimental security analysis of a modern automobile," in *Proc. IEEE Symp. Secur. Privacy*, 2010, pp. 447–462.

[3] M. Enev, A. Takakuwa, K. Koscher, and T. Kohno, "Automobile driver fingerprinting," *Proc. Privacy Enhancing Technol.*, vol. 2016, no. 1, pp. 34–50, 2016.

[4] A. Hazem and H. Fahmy, "LCAP-A lightweight can authentication protocol for securing in-vehicle networks," *Proc. 10th Escar Embedded Secur. Cars Conf.*, vol. 6, p. 172, 2012.

[5] C. Miller and C. Valasek, "Remote exploitation of an unaltered passenger vehicle," *Black Hat USA*, vol. 2015, no. S 91, 2015.

[6] S. Woo, H. J. Jo, and D. H. Lee, "A practical wireless attack on the connected car and security protocol for in-vehicle CAN," *IEEE Trans. Intell. Transp. Syst.*, vol. 16, no. 2, pp. 993–1006, Apr. 2015.

[7] C. Beek and R. Samani, "DEFCON – Connected car security," 2017. Accessed: Feb. 25, 2021. [Online]. Available: https://www.mcafee.com/blogs/other-blogs/mcafee-labs/defcon-connected-car-security/

[8] R. Ross, V. Pillitteri, R. Graubart, D. Bodeau, and R. McQuaid, "Developing cyber resilient systems: A systems security engineering approach," NIST, Gaithersburg, MD, Tech. Rep. SP 800–160, 2019.

[9] G. Bella, P. Biondi, G. Costantino, and I. Matteucci, "TOUCAN: A protocol to secure controller area network," in *Proc. ACM Workshop Automot. Cybersecurity*, 2019, pp. 3–8.

[10] A.-I. Radu and F. D. Garcia, "LeiA: A lightweight authentication protocol for CAN," in *Proc. Eur. Symp. Res. Comput. Secur.*, 2016, pp. 283–300.

[11] P.-S. Murvay and B. Groza, "TIDAL-CAN: Differential timing based intrusion detection and localization for controller area network," *IEEE Access*, vol. 8, pp. 68 895–68 912, 2020.

[12] M. Foruhandeh, Y. Man, R. Gerdes, M. Li, and T. Chantem, "SIMPLE: Single-frame based physical layer identification for intrusion detection and prevention on in-vehicle networks," in *Proc. 35th Annu. Comput. Secur. Appl. Conf.*, 2019, pp. 229–244.

[13] M. Kneib, O. Schell, and C. Huth, "EASI: Edge-based sender identification on resource-constrained platforms for automotive networks," in *Proc. Netw. Distrib. Syst. Secur. Symp.*, 2020, pp. 1–16.
[14] B. W. Lampson, "A note on the confinement problem," *Commun. ACM*, vol. 16, no. 10, pp. 613–615, Oct. 1973.
[15] Z. Wu, Z. Xu, and H. Wang, "Whispers in the hyper-space: High-bandwidth and reliable covert channel attacks inside the cloud," *IEEE/ACM Trans. Netw.*, vol. 23, no. 2, pp. 603–615, Apr. 2015.
[16] M. Lipp *et al.*, "Meltdown: Reading kernel memory from user space," in *Proc. 27th USENIX Secur. Symp.*, 2018, pp. 973–990.
[17] X. Ying, G. Bernieri, M. Conti, and R. Poovendran, "TACAN: Transmitter authentication through covert channels in controller area networks," in *Proc. 10th ACM/IEEE Int. Conf. Cyber-Physical Syst.*, 2019, pp. 23–34.
[18] B. Groza, L. Popa, and P.-S. Murvay, "Incanta-intrusion detection in controller area networks with time-covert authentication," in *Proc. Secur. Saf. Interplay Intell. Softw. Syst.*, 2018, pp. 94–110.
[19] B. Groza, L. Popa, and P.-S. Murvay, "CANTO-covert authentication with timing channels over optimized traffic flows for CAN," *IEEE Trans. Inf. Forensics Security*, vol. 16, pp. 601–616, Aug. 2020.
[20] B. Groza, L. Popa, and P. Murvay, "Tricks–time triggered covert key sharing for controller area networks," *IEEE Access*, vol. 7, pp. 104 294–104 307, 2019.
[21] A. Mueller and T. Lothspeich, "Plug-and-secure communication for CAN," *CAN Newslett.*, vol. 4, pp. 10–14, 2015.
[22] S. Soderi, L. Mucchi, M. Hämäläinen, A. Piva, and J. H. Iinatti, "Physical layer security based on spread-spectrum watermarking and jamming receiver," *Trans. Emerg. Telecommun. Technol.*, vol. 28, no. 7, 2017, Art. no. e3142.
[23] D. A. Wagner and S. M. Bellovin, "A "bump in the stack" encryptor for ms-dos systems," in *Proc. Symp. Netw. Distrib. Syst. Secur.*, 1996, pp. 155–160.
[24] S. Soderi, "Acoustic-based security: A key enabling technology for wireless sensor network," *Int. J. Wireless Inf. Netw.*, vol. 27, no. 1, pp. 45–59, Nov. 2019.
[25] S. Soderi, "Enhancing security in 6G visible light communications," in *Proc. 2nd 6G Wireless Summit*, 2020, pp. 1–5.
[26] Road vehicles – Interchange of digital information – Controller area network (CAN) for high-speed communication, International Organization for Standardization, Standard ISO 11898: 1993, 1993.
[27] Road vehicles – Controller area network (CAN) – Part 2: High-speed medium access unit, International Organization for Standardization, Standard ISO 11898–2:2016, 2016.
[28] Road vehicles – Controller area network (CAN) – Part 3: Low-speed, fault-tolerant, medium-dependent interface, International Organization for Standardization, Standard ISO 11898–3:2006, 2006.
[29] Road vehicles – Controller area network (CAN) – Part 1: Data link layer and physical signalling, International Organization for Standardization, Standard ISO 11898–1:2015, 2015.
[30] M. Hanspach and M. Goetz, "On covert acoustical mesh networks in air," 2014 , *arXiv:1406.121*.
[31] F. A. P. Petitcolas, R. J. Anderson, and M. G. Kuhn, "Information hiding-a survey," *Proc. IEEE*, vol. 87, no. 7, pp. 1062–1078, Jul. 1999.
[32] X. Li, C. Yu, M. Hizlan, W.-T. Kim, and S. Park, "Physical layer watermarking of direct sequence spread spectrum signals," in *Proc. IEEE Mil. Commun. Conf.*, 2013, pp. 476–481.
[33] I. J. Cox, J. Kilian, F. Leighton, and T. Shamoon, "Secure spread spectrum watermarking for multimedia," *IEEE Trans. Image Process.*, vol. 6, no. 12, pp. 1673–1687, Dec. 1997.
[34] C. Shannon, "Communication theory of secrecy systems," *Bell Syst. Tech. J.*, vol. 28, no. 4, pp. 656–715, Oct. 1949.
[35] A. Van Herrewege, D. Singelee, and I. Verbauwhede, "CANAuth–A simple, backward compatible broadcast authentication protocol for CAN bus," in *Proc. ECRYPT Workshop Lightweight Cryptogr.*, 2011, Art. no. 20.
[36] 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 *Proc. IEEE Veh. Technol. Conf.*, 2011, pp. 1–5.
[37] B. Groza, S. Murvay, A. van Herrewege, and I. Verbauwhede, "LiBrA-CAN: A lightweight broadcast authentication protocol for controller area networks," in *Proc. Int. Conf. Cryptol. Netw. Secur.*, 2012, pp. 185–200.
[38] R. Kurachi, Y. Matsubara, H. Takada, N. Adachi, Y. Miyashita, and S. Horihata, "CaCAN-centralized authentication system in can (controller area network)," in *Proc. 14th Int. Conf. Embedded Secur. Cars*, 2014, pp. 1–10.
[39] Q. Wang and S. Sawhney, "VeCure: A practical security framework to protect the CAN bus of vehicles," in *Proc. Int. Conf. Internet Things*, 2014, pp. 13–18.
[40] S. Nürnberger and C. Rossow, "– vatican – vetted, authenticated CAN bus," in *Proc. Cryptographic Hardware Embedded Syst.*, 2016, pp. 106–124.
[41] J. Van Bulck, J. T. Mühlberg, and F. Piessens, "VulCAN: Efficient component authentication and software isolation for automotive control networks," in *Proc. 33rd Annu. Comput. Secur. Appl. Conf.*, 2017, pp. 225–237.
[42] S. Fassak, Y. E. H. El Idrissi, N. Zahid, and M. Jedra, "A secure protocol for session keys establishment between ECUs in the CAN bus," in *Proc. Int. Conf. Wireless Netw. Mobile Commun.*, 2017, pp. 1–6.
[43] B. Palaniswamy, S. Camtepe, E. Foo, and J. Pieprzyk, "An efficient authentication scheme for intra-vehicular controller area network," *IEEE Trans. Inf. Forensics Security*, vol. 15, pp. 3107–3122, Mar. 2020.
[44] Y. Xiao, S. Shi, N. Zhang, W. Lou, and Y. T. Hou, "Session key distribution made practical for CAN and CAN-FD message authentication," in *Proc. Annu. Comput. Secur. Appl. Conf.*, 2020, pp. 681–693.
[45] H. Mun, K. Han, and D. H. Lee, "Ensuring safety and security in CAN-based automotive embedded systems: A combination of design optimization and secure communication," *IEEE Trans. Veh. Technol.*, vol. 69, no. 7, pp. 7078–7091, Jul. 2020.
[46] B. Groza, L. Popa, and P.-S. Murvay, "Highly efficient authentication for CAN by identifier reallocation with ordered CMACs," *IEEE Trans. Veh. Technol.*, vol. 69, no. 6, pp. 6129–6140, Jun. 2020.
[47] T.-Y. Youn, Y. Lee, and S. Woo, "Practical sender authentication scheme for in-vehicle CAN with efficient key management," *IEEE Access*, vol. 8, pp. 86 836–86 849, 2020.
[48] M. D. Pesé, J. W. Schauer, J. Li, and K. G. Shin, "S2-CAN: Sufficiently secure controller area network," in *Proc. Annu. Comput. Secur. Appl. Conf.*, 2021, pp. 425–438.
[49] T. Limbasiya, A. Ghosal, and M. Conti, "AutoSec: Secure automotive data transmission scheme for in-vehicle networks," in *Proc. 23rd Int. Conf. Distrib. Comput. Netw.*, 2022, pp. 208–216.
[50] H. J. Jo and W. Choi, "A survey of attacks on controller area networks and corresponding countermeasures," *IEEE Trans. Intell. Transp. Syst.*, to be published, doi: 10.1109/TITS.2021.3078740.
[51] B. Groza, L. Popa, P.-S. Murvay, Y. Elovici, and A. Shabtai, "CANARY-a reactive defense mechanism for controller area networks based on active RelaYs," in *Proc. 30th USENIX Secur. Symp.*, 2021, pp. 4259–4276.
[52] S. U. Sagong, X. Ying, A. Clark, L. Bushnell, and R. Poovendran, "Cloaking the clock: Emulating clock skew in controller area networks," in *Proc. ACM/IEEE 9th Int. Conf. Cyber-Physical Syst.*, 2018, pp. 32–42.
[53] J. Zhou, P. Joshi, H. Zeng, and R. Li, "BTMonitor: Bit-time-based intrusion detection and attacker identification in controller area network," *ACM Trans. Embedded Comput. Syst.*, vol. 18, no. 6, pp. 1–23, 2019.
[54] W. Choi, K. Joo, H. J. Jo, M. C. Park, and D. H. Lee, "VoltageIDS: Low-level communication characteristics for automotive intrusion detection system," *IEEE Trans. Inf. Forensics Secur.*, vol. 13, no. 8, pp. 2114–2129, Aug. 2018.
[55] X. Ying, G. Bernieri, M. Conti, L. Bushnell, and R. Poovendran, "Covert channel-based transmitter authentication in controller area networks," *IEEE Trans. Dependable Secure Comput.*, to be published, doi: 10.1109/TDSC.2021.3068213.
[56] S. Soderi and R. De Nicola, "6G networks physical layer security using RGB visible light communications," *IEEE Access*, vol. 10, pp. 5482–5496, 2022.
[57] J. G. Proakis, *Digital Communications*, 4th ed. Boston, MA, USA: McGraw-Hill, 2000. [Online]. Available: http://www.loc.gov/catdir/description/mh021/00025305.html
[58] H. Malvar and D. Florencio, "Improved spread spectrum: A new modulation technique for robust watermarking," *IEEE Trans. Signal Process.*, vol. 51, no. 4, pp. 898–905, Apr. 2003.
[59] A. Palanca, E. Evenchick, F. Maggi, and S. Zanero, "A stealth, selective, link-layer denial-of-service attack against automotive networks," in *Proc. Int. Conf. Detection Intrusions Malware, Vulnerability Assessment*, 2017, pp. 185–206.

[60] S. Vanderhallen, J. Van Bulck, F. Piessens, and J. T. Mühlberg, "Robust authentication for automotive control networks through covert channels," *Comput. Netw.*, vol. 193, 2021, Art. no. 108079.
[61] S. Jain and J. Guajardo, "Physical layer group key agreement for automotive controller area networks," in *Proc. Int. Conf. Cryptographic Hardware Embedded Syst.*, 2016, pp. 85–105.

Simone Soderi (Senior Member IEEE) received the MSc degree from the University of Florence, Italy, in 2002, and the DrSc degree from the University of Oulu, Finland, in 2016. He has more than fifteen years of experience in embedded systems and safety-critical architectures. His skills range from cybersecurity and wireless communications to software engineering. During 2010-2014 he was a member of the Steering Committee of a joint research project between General Electric, Italy, and the Centre for Wireless Communications, University of Oulu, Finland. In 2011-2015, he contributed to ETSI for ultra-wideband devices in road and rail vehicles. After the doctoral degree, he was appointed as Italy Cybersecurity manager with Alstom, Florence, Italy. In 2019-2020, he was the national coordinator of the CyberChallenge.IT project. Currently, he is a assistant professor with IMT School for Advanced Studies Lucca. His research topics include cybersecurity for critical infrastructure systems, 6G, covert channels, network security, physical layer security, electromagnetic emissions security, VLC, and UWB. He has been TPC member of several conferences and served as a reviewer of *IEEE Transaction on Intelligent Transport Systems*, *IEEE Wireless Communications Magazine*, PIMRC, and INFOCOM. He published journal-conference papers and chapters in a book. He holds five patents regarding wireless communications and positioning.

Riccardo Colelli received the Laurea degree in management and automation engineering from the University of Roma Tre, Rome, Italy, in 2018, and the PhD degree in computer science and automation from the University of Roma Tre, Rome, Italy. His current research interests include cyber physical systems, cybersecurity and critical infrastructure protection.

Federico Turrin received the master's degree in computer engineering from the University of Padova, Italy, in 2019, where he is currently working toward the interdisciplinary PhD degree in brain, mind, and computer science, since October 2019. He has been a visiting researcher with SUTD Singapore, in 2022. His research interests lie primarily in cyber-physical system security with a particular focus on industrial control systems security, vehicles security, and application of machine learning techniques for anomaly detection.

Federica Pascucci (Senior Member, IEEE) received the MS degree in computer science and automation engineering from the University of Roma Tre, Rome, Italy, in 2000, and the PhD degree in system engineering from the University of Rome La Sapienza, Rome, in 2004. Since 2020, she has been an associate professor with the Department of Engineering, University of Roma Tre. She was a visiting scholar with the University of ÖRebro, ÖRebro, Sweden, in 2003 and the University of Cyprus, Cyprus, in 2013. She is the principal investigator in several EU funded projects. She has published more than 100 journal and conference papers. Her current research interests include wireless sensor networks, indoor localization, cyber-physical systems, industrial control systems, and critical infrastructure protection.

Mauro Conti (Fellow, IEEE) received the PhD degree from the Sapienza University of Rome, Italy, in 2009. He is a full professor with the University of Padua, Italy. He is also affiliated with TU Delft and University of Washington, Seattle. After his PhD, he was a post-doc researcher with Vrije Universiteit Amsterdam, The Netherlands. In 2011, he joined as assistant professor with the University of Padua, where he became associate professor, in 2015, and full professor, in 2018. He has been a visiting researcher with GMU, UCLA, UCI, TU Darmstadt, UF, and FIU. He has been awarded with a Marie Curie Fellowship, 2012 by the European Commission, and with a Fellowship by the German DAAD, 2013. His research is also funded by companies, including Cisco, Intel, and Huawei. His main research interest is in the area of security and privacy. In this area, he published more than 400 papers in topmost international peer-reviewed journals and conferences. He is editor-in-chief for *IEEE Transactions on Information Forensics and Security*, area editor-in-chief for *IEEE Communications Surveys & Tutorials*, and has been associate editor for several journals, including *IEEE Communications Surveys & Tutorials*, *IEEE Transactions on Dependable and Secure Computing*, *IEEE Transactions on Information Forensics and Security*, and *IEEE Transactions on Network and Service Management*. He was program chair for TRUST 2015, ICISS 2016, WiSec 2017, ACNS 2020, CANS 2021, and general chair for SecureComm 2012, SACMAT 2013, NSS 2021 and ACNS 2022. He is senior member of the ACM, and fellow of the Young Academy of Europe.

Open Access funding provided by 'Scuola IMT Alti Studi Lucca' within the CRUI CARE Agreement