IEEE Access
Multidisciplinary | Rapid Review | Open Access Journal

Date of publication xxxx 00, 0000, date of current version xxxx 00, 0000.

Digital Object Identifier 10.1109/ACCESS.2019.DOI

TRICKS - time TRIggered Covert Key Sharing for Controller Area Networks

BOGDAN GROZA[1], LUCIAN POPA[2] AND PAL-STEFAN MURVAY.[3]

[1]Faculty of Automatics and Computers, Politehnica University of Timisoara, 300223 Timisoara, Romania (e-mail: bogdan.groza@aut.upt.ro)
[2]Faculty of Automatics and Computers, Politehnica University of Timisoara, 300223 Timisoara, Romania (e-mail: lucian.popa@aut.upt.ro)
[3]Faculty of Automatics and Computers, Politehnica University of Timisoara, 300223 Timisoara, Romania (e-mail: pal-stefan.murvay@aut.upt.ro)

Corresponding author: Bogdan Groza (e-mail: bogdan.groza@aut.upt.ro).

This work was supported by a grant of Ministry of Research and Innovation, CNCS-UEFISCDI, project number PN-III-P1-1.1-TE-2016-1317, within PNCDI III (2018-2020).

ABSTRACT There are dozens of proposals for securing the Controller Area Network (CAN), however, only a few of them are concerned on how to share secret keys between CAN nodes. Recently, some works have used the non-destructive property of CAN arbitration in order to exchange a secret key and achieve information theoretic security for the key exchange. In our proposals we exploit both delays and the non-destructive arbitration of CAN to achieve a secure key exchange. While our approach is less efficient when it comes to bandwidth, we do not require any kind of additional hardware and we build our implementation on the software layer which is accessible for any CAN based application. To boost efficiency, we finally bootstrap secret keys by means of the guessing-resilient protocols such as EKE (Encrypted-Key-Exchange) and SPEKE (Simple Password Exponential Key Exchange). In principle a few dozen frames suffice for a secure key-exchange between two CAN nodes. We discuss several protocol versions and extensions for the case of more than two parties. We also present experimental results on modern automotive-grade controllers to prove the performance of our solution.

INDEX TERMS Authentication, Cryptography, Microcontrollers, Network security

I. INTRODUCTION AND MOTIVATION

In the recent years, numerous works have been focusing on designing security protocols for the CAN bus [18], showing innovative solutions from the use of cryptographic message authentication codes [20], [27], group key-sharing [17], signal characteristics [35], [11] or network delays [9], [37]. Not surprisingly, the majority of these solutions, e.g., works in [27], [20], [41] and many others, are based on cryptographic Message Authentication Codes (MACs) that do require a secret shared key. However, with the exception of the work in [34] later explored in a more comprehensive way by [23], little focus was put on how to exchange cryptographic keys on the CAN bus.

In an exceptional engineering work, Mueller and Lothspeich [34] show how to exploit the dominant vs. recessive state of the CAN bus in order to securely exchange a key. On the CAN bus, two or more nodes can write bits at the same time and a 0 will always overwrite a 1. The principle they use is simple and effective: two CAN nodes generate a random sequence of bits and then start to place the bits simultaneously on the bus. If both bits are 1 then the bus will be in a recessive state and the adversary will know that both nodes have generated a recessive bit. However, if the pair of bits generated on the nodes are $(0,0), (0,1)$ or $(1,0)$ the bus will have a dominant level in all cases and the adversary cannot discern between the three cases. For genuine nodes however, the node that generated the 1 will know that the other node has generated a 0 if the bus is dominant and thus one bit is extracted. For the node that generated a 0, the $(0,0)$ and $(0,1)$ case are again indistinguishable but this situation is solved by re-sending the complement value of the bits. This again discloses the $(0,0)$ pair to the adversary since this changes to $(1,1)$ and the bus is recessive. However, the $(0,1)$ and $(1,0)$ cases remain indistinguishable for an adversary and from these, the genuine nodes are able to extract one bit for the key (since each node is aware of the bit it placed on the bus).

Contribution and addressed scenario. Figure 1 is a basic depiction of the CAN bus and of our addressed scenario. The depiction is also suggestive with respect to one of the protocol designs that we later address. For this reasons frames are depicted arriving at intervals Δ and each node

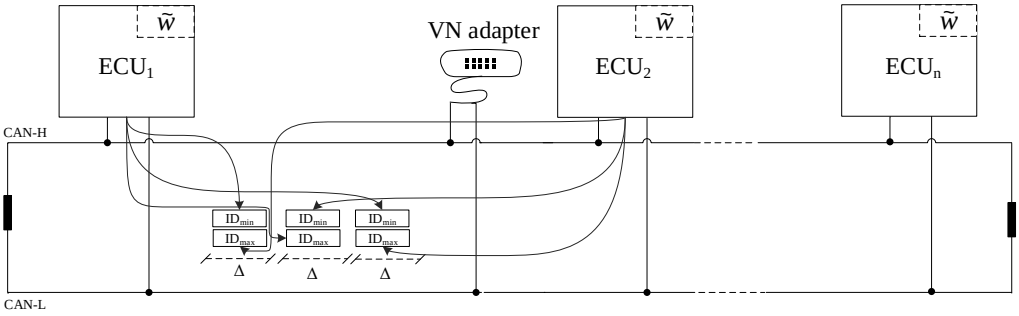

FIGURE 1. Structure of the CAN bus and addressed scenario.

sends an ID that is the minimum or the maximum from the particular time interval (more on this later). We also depict a VN adapter which is an industry-standard instrument from Vector and was used by us mostly for debugging purposes. We assume that a weak shared-secret $\widetilde{w}$ is already shared by the nodes ECU_1, ECU_2, ..., ECU_n. Without the existence of such a secret, security can be achieved only with respect to a weaker form of adversaries (that are unable to unplug nodes from the bus as we later discuss). We consider that it is realistic to assume that in-vehicle Electronic Control Units (ECU) do share a weak secret key of some sort. For example, following the fundamental resurrecting-duckling paradigm [38], components inside the car may be imprinted with a secret derived from the first seconds when the car is start for the first time. This hopefully happens in a secure environment at the manufacturer or seller. We do not require for this secret to be a strong cryptographic key, but rather a weak one which is sufficient to bootstrap a secure cryptographic key later. Nonetheless, the secret $\widetilde{w}$ can be an existing secret shared between the nodes as current AUTOSAR specifications have support for key-exchange protocols [2] and ask for security on in-vehicle units [3]. In this case, the procedures that we discuss here can be used as a covert channel to further re-enforce an existing key.

In this setting, we design a solution that can be fully implemented at the application layer without requiring any modifications of the CAN protocol stack. That is, the ability to program timer-counter circuits and to send CAN frames is enough to negotiate a session key in a secure manner. We later add some cryptographic flavors to our protocol suite in order to boost efficiency in the initial versions of our protocol. We discuss four main variations based on data or remote frames, identifier priority and nonetheless timings. The last two versions of our schemes: time-triggered minimax and the randomized delay key negotiation set room for piggybacking frames with parts of the keys that are shared via the Diffie-Hellman (DH) version of the Encrypted-Key-Exchange protocol (EKE) [6] and Simple Password Exponential Key Exchange (SPEKE) [22]. This is possible in case of the later schemes that we introduce since the bits of the key that are to be negotiated are known in an a-priori manner.

The EKE-DH protocol is known to be secure against guessing attacks and thus we can bootstrap a session key by using a small entropy secret for authenticating the larger session key. For nodes that do not have enough computation power to rely on the Diffie-Hellman key-exchange, all the schemes that we introduce are functional even without this cryptographic extension of our protocol. Thus we try to cover both scenarios where mid to high-end automotive grade controllers are present, as well as scenarios with low-end cores.

II. BACKGROUND AND EXPERIMENTAL SETUP

We begin with a brief background on CAN then we revisit some of the existing related work. Finally, we give some details on our experimental setup.

A. CAN BASICS

CAN is the de facto standard for communication in automotive in-vehicle networks. Bit rates of up to 1Mbit/s can be used on CAN, however, actual implementations only go up to 500Kbit/s to assure reliability. At the physical layer CAN is implemented as a two wire differential bus which uses two levels to encode transmissions: a dominant and a recessive level. A dominant level (interpreted as logical "0") is asserted when at least one CAN node actively drives the bus, while the recessive level (interpreted as logical "1") is the result of none of the nodes driving the bus. Therefore, a dominant level will always overwrite a recessive level.

The main part of CAN communication consists of standard and/or extended data frames, depicted in Figure 2, which can carry up to 8 bytes of data. The difference between the two types of data frames consists in the identifier field which is 11 bits long in standard frames and 29 bits in extended frames.

Remote request frames differ from data frames by the RTR bit which should be set to "1" (recessive) in the case of remote frames. This type of frames is used to request the transmission of a specific data frame.

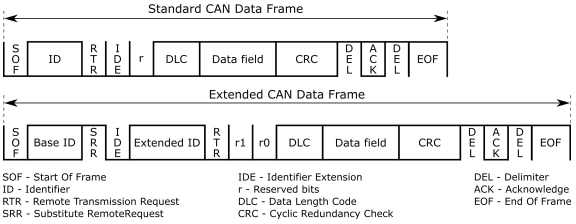

FIGURE 2. Structure of standard and extended CAN frames.

B. RELATED WORK

Following the attacks reported on modern cars, e.g., [8], [31], [32], numerous proposals for assuring security on in-vehicle buses started to appear. As expected, the largest body of proposals is centered around embedding cryptographic message authentication codes (MAC), e.g., [20], [4] and some go further to optimal traffic allocation to cope with security demands [28], [29], [42] or analyze trade-offs between safety and security [13].

Blameless, while the use of MACs in all previous works mandates for a shared secret key, very few research proposals have actually focused on how to share cryptographic keys on the CAN bus. This task is more complicated than what may appear on the first sight. For example, the adoption of out-of-the-box public-key exchanges based on the RSA [36] or Diffie-Hellman [14] is problematic due to the large public-keys or by the requirement of the corresponding public-key infrastructure (PKI), certificates, etc.

The procedure introduced by Mueller and Lothspeich [34] was the first to use intrinsic properties of the CAN bus for providing key agreement. The scheme was later extended by Jain and Guajardo to achieve group key agreement over CAN [23]. In a more recent work [24] the authors evaluate the resilience of this type of CAN key agreement mechanism to probing attacks. As it turns out, differences in signaling behavior specific to each node can be used to identify the value of the bit generated by each of the nodes participating in the key agreement scheme. Vulnerability to probing attacks can be alleviated by the use of mechanisms that mask the unique node signaling behavior as proposed by [24]. Particularities of the physical layer, voltage level in particular, have in fact become recent preoccupation for many works focused on intrusion detection on the CAN bus, e.g., [35], [10], [11], [26]. Other works used physical properties of oscillators to distinguish between nodes based on clock drifts, i.e., [9]. But recent research has also shown that clock skews can be mimicked and impersonation is possible [37]. Our work uses both the wired-and behaviour of the CAN bus as well as delays in order to negotiate a secret key. While probing attacks and delays may reveal some information about the senders, countermeasures exists, e.g., nodes sending in parallel [24] or forcing small random delays to hide the real clock drift.

Properties of the physical layer were also used for key establishment in wireless networks [30] which is an emerging sector of in-vehicle communication. In wireless communication, the principle of reciprocity of random fading wireless channels is employed for key generation. While such an approach is entirely distinct to ours, the protocols that we describe here can be ported to some wireless networks as well. This however may subject of future work and it is out of reach for the current publication.

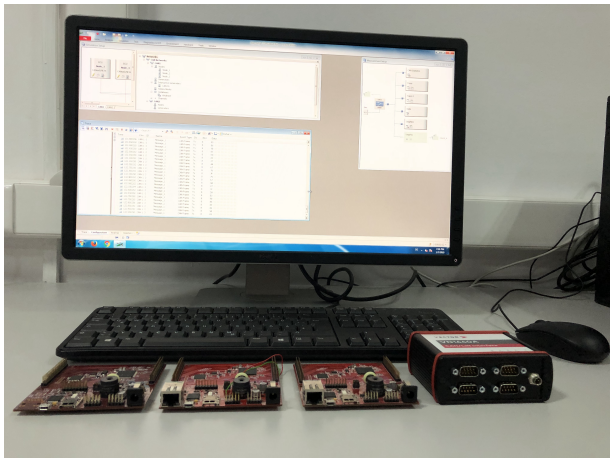

FIGURE 3. Hardware and software components employed in our experiments.

C. SETUP FOR OUR EXPERIMENTS

In brief, our setup includes several development boards equipped with high-performance automotive grade microcontrollers from the Infineon Aurix family which we interface to the CANoe environment running on a standard PC via a VN adapter from Vector (for monitoring frame arrival on the bus). The components employed in our experimental setup are depicted in Figure 3.

The experiments described in this paper have been performed by pairing nodes implemented on AURIX-based development boards. We employed three AURIX Application Kit models from Infineon which differ in the microcontroller version used:

- The TC224 features one core running at a frequency of up to 133MHz along with 96kB of RAM and 1MB of FLASH.
- The TC277 is equipped with three cores, each of which can operate at 200MHz, as well as 472kB RAM and 4MB FLASH.
- The TC297 microcontroller as one of the top performers in the AURIX class comes with 728kB RAM, 8MB FLASH and three cores that can be clocked at 300MHz.

A VN PC CAN adapter from Vector was used to interface between the bus connecting the node pairs and the CANoe analysis environment for communication monitoring and trace recording on a PC. Further analysis of the recorded traces was performed in the Mathematica environment.

III. TIME-TRIGGERED KEY-EXCHANGE PROTOCOLS

We discuss four protocol versions in what follows. We start from using remote vs. data frames, overlapping IDs, then we

TABLE 1. Summary of notations

ID	identifier field of a CAN message
ECU	Electronic Control Unit
Pr_{bad}	probability that a frame discloses the corresponding bit
$\mathsf{H}_{average}$	mean entropy (as function of k frames)
T	duration of the key exchange (as function of k frames)
Δ	estimated delay between two frames
$b_{\blacklozenge}$	array of random bits generated by $\blacklozenge \in \{A, B\}$
$\neg b_{\blacklozenge}$	complement of values in previous bit array $\blacklozenge \in \{A, B\}$
$x_{\blacklozenge}$	random delay generated by $\blacklozenge \in \{A, B\}$
$id_{\blacklozenge}$	array of random ID generated by $\blacklozenge \in \{A, B\}$
$\mathsf{ID}_{\mathsf{min}}$	the ID with the minimum value (whenever two IDs occur at the same time)
$\mathsf{ID}_{\mathsf{max}}$	the ID with the maximum value (whenever two IDs occur at the same time)

add timing and finally we include a randomized distribution of the delays for sender and receiver node.

A. SUMMARY OF NOTATIONS

Table 1 gives a summary on the notations that we use for the following protocol schemes. To quantify the performance of the protocol we mainly use three metrics: the probability that a frame is bad Pr_{bad} by which we mean that the bit disclosed by the frame is visible to an adversary, the mean entropy $\mathsf{H}_{average}$ which is a function of the number of frames sent by one node and the total time T for the key-exchange protocol.

B. ADVERSARY MODEL

We do consider the general case of a Dolev-Yao adversary [15] that has full control over the communication, i.e., he can read, write or stop messages on the communication channel at will.

The first schemes that we present for key-exchange based on covert timing channels are not fully secure against such an adversary. That is, if an adversary is able to unplug a node from the bus and send messages instead of the genuine node, then such an impersonation will be successful since there is no mechanism to discern between such an adversary and the genuine node. Previous proposal, such as the Mueller and Lothspeich protocol [34], will be vulnerable to such an adversary as well. We do emphasize however, that such an attack will not be easy to mount since it is hard to unplug nodes at will inside a car. Previous research has focused on DoS (Denial of Service) attacks that will place CAN nodes in bus-off, e.g., [9]. But the implied procedures are not yet at the point of giving the adversary an on-off that will disconnect/connect nodes from the bus at will, e.g., CAN nodes may stay in bus-off for a very short amount of time or not enter in bus-off state at all. If the adversary can also inject or modify messages (without being able to disconnect nodes from the bus), then the attack will result in a DoS since the genuine nodes will end up with wrong keys. So we consider that even the first schemes are secure in front of a more limited adversary that can only eavesdrop on the channel. For the case of a stronger adversary, that has full control over the nodes, we later design a protocol that uses a weak existing secret along with the initial version of our schemes to securely exchange a session key based on guessing resilient protocol Encrypted Key Exchange (EKE) [6], [5]. To reduce the overhead we rely on elliptical-curves and rely on the main idea from the SPEKE protocol [22] also taking into account recent attacks and modifications proposed in [19].

C. PRELIMINARY IMPLEMENTATION NOTES

We now give some implementation details that are common for the implementation of all protocols that we introduce in the next sub-section. For a more conclusive presentation, starting from the next section, we support the theoretical description of the schemes with practical results (experimental data from the CAN bus). For clarity, we summarize here some preliminary implementation notes.

For implementing the timing functionality required by the proposed key negotiation mechanisms we employed the Capture/Compare Unit 6 Timer (CCU6) module available on the AURIX chips. The T13 timer block of the CCU6 is configured to generate periodic interrupts which represent the system tick on which the local timestamping functionality is based.

The CAN communication on the embedded platforms side is assured through the on-chip MultiCAN+ module. It handles message transmission and reception through so called message objects which serve as storage for sent and received frames and can be used for message filtering. A maximum of 256 or 128 message objects are available on each chip depending on the microcontroller variant. We configured message objects to support up to 64 different send and receive message IDs and set the acceptance filter to 0 in order to process any received CAN frame regardless of the frame identifier. The bit rate of the CAN controller was configured to 500Kbit/s according to the SAE Recommended Practice SAE-J2284-3.

In all of the described protocols, after a startup phase in which hardware initialization and configuration takes place, each node waits for a CAN frame that triggers the initialization of the timer. This frame is sent from CANoe. The timer interrupt period is configured at 2 μs.

After the trigger-frame is received, the software is configured in order to send a CAN Frame (remote or standard depending on the experiment) at a specific timestamp, measured by the timer. After every data transfer, each node saves the ID and the timestamp and also verifies if it has received any CAN frame before and if so, it saves the ID of the frame and the timestamp. After the number of expected frames is received, each node computes the other node's random bits based on the timestamp arrays. Otherwise, every node computes all the bit values based on the timestamp arrays.

D. COVERT KEY-EXCHANGE PROCEDURES

Data vs. Remote Frame Negotiation. This type of key negotiation is identical to Mueller and Lothspeich [34] principle except that is far more inefficient since a single frame is required for a single bit. Discussing the principles is however

relevant for the protocols that follow and also for clarifying how Mueller and Lothspeich solution works [34].

An array of bits is randomly generated on each node, i.e., $b_A = b_A^1, b_A^2, ..., b_A^k, b_B = b_B^1, b_B^2, ..., b_B^k$. Each bit in the arrays establishes if the frame that needs to be sent by the corresponding node is a data frame (bit equals to 0) or a remote frame (bit equals to 1). Both nodes A and B broadcast the frame simultaneously at intervals Δ. That is, the i-th frame from each node is sent at time $t_{start} + i\Delta$ (assuming communication has start at t_{start}). The value of Δ should be chosen so that it accommodates the transmission time of a CAN frame at the current baud-rate. For example, in our experiments with a 500Kbit/s bus, Δ was set to $200\mu s$ (this value can be further reduced to the maximum duration of a frame on the bus).

Figure 4 shows the situations that may occur on the bus. If the bits on the nodes are the complement of each other, i.e., the 01 and 10 cases (i), (ii), a data frame will appear on the bus. The node that sent the remote frame knows that the other node has placed the data frame. However, the node that sent the data frame does not know whether the other node sent a remote frame or a data frame, i.e., case (ii) vs. (iii) for node B, since in both situations a data frame will occur on the bus. This is further clarified when the complement of the bit-array is sent, i.e., $\neg b_A, \neg b_B$. In case (iv) when both nodes have sent a remote frame this is visible to the adversary so the security of the bit is compromised. The ID that is broadcast is a fixed value for all nodes. We also note that transmission errors may occur if there is no acknowledgement bit and if only the two sender nodes are present on the bus the sender cannot place an acknowledgement bit (according to CAN specifications). To avoid nodes entering in bus-off mode we configured the Infineon CAN module not to re-attempt transmission. If a third node is present on the bus (the most likely) then such a situation will not occur since the third node will place the acknowledgement bit.

In this case, obviously, half of the bits are visible to the adversary, i.e., cases (iv) in Figure 4 and the complement of case (iii), and thus $\mathsf{Pr}_{bad} = 1/2$. Consequently, only half of the frames from each will contribute to the entropy of the key which leads to $\mathsf{H}_{average}(k) = (1 - \mathsf{Pr}_{bad})\, k = k/2$. The runtime of the protocol is the time to send k frames by each node and then their complements resulting in $\mathsf{T}(k) = 2k\Delta$.

Experimental results on this approach now follow. In all experiments two nodes are present on the bus and send messages according to the previous protocol description. For simplicity, the figures show only data for the first 32 frames. In Figure 5 we show frames on each of the two nodes placed on the X-axis according to the arrival timestamp. The Y-axis represents the value of the ID in decimal. Whenever a frame is remote, to avoid overlapping with data frame, the ID is represented as $-ID$, thus remote frames are depicted by negative IDs. The IDs broadcasted by the nodes in each time interval are identical and this is a assured by using a non-secret seed for a pseudo-random number generator. Finally, in Figure 6 we show frames as they arrived on the bus by

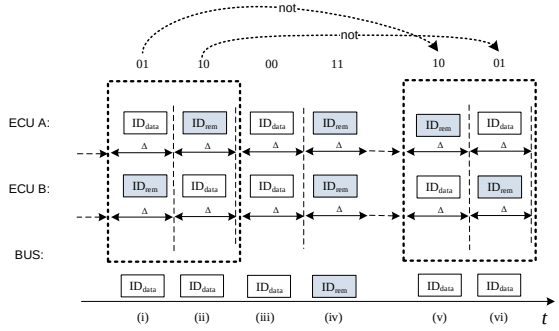

FIGURE 4. Data vs. Remote frame negotiation (based on Mueller and Lothspeich principle [34]).

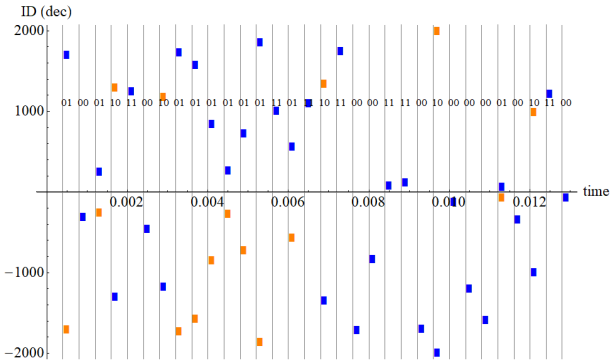

FIGURE 5. Data vs. Remote frame negotiation: frames on node A and node B (orange vs. blue).

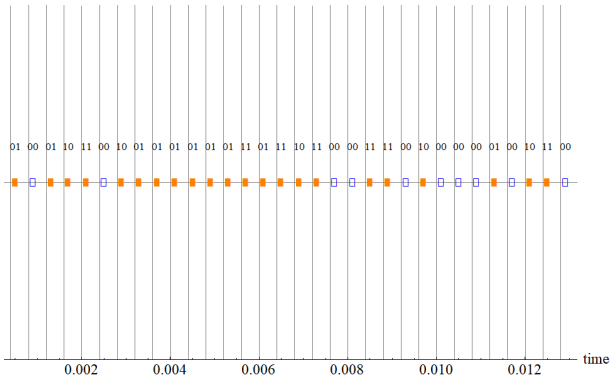

FIGURE 6. Data vs. Remote frame negotiation, frames arrived on the bus data (square) vs. remote (empty square).

using the trace from CANoe. Remote frames (which appear in case when both nodes are sending a remote frame) are depicted by an empty square and data frames by a full square. Figure 7 summarizes the experiments by placing a check-mark over the frames that arrived at the expected time, this holds for all frames.

Minimax Negotiation. The minimax key exchange uses randomized IDs that are sent by each node. Each node,

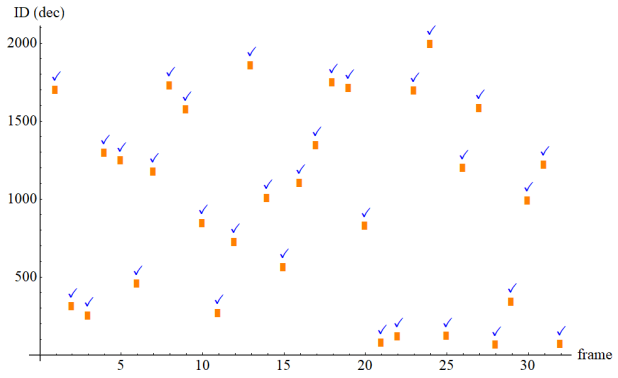

FIGURE 7. Data vs. Remote frame negotiation: expected arrival frame (as checkmark) vs. arrived frame (square).

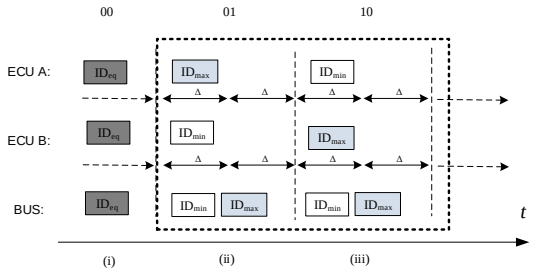

FIGURE 8. Minimax Key Exchange.

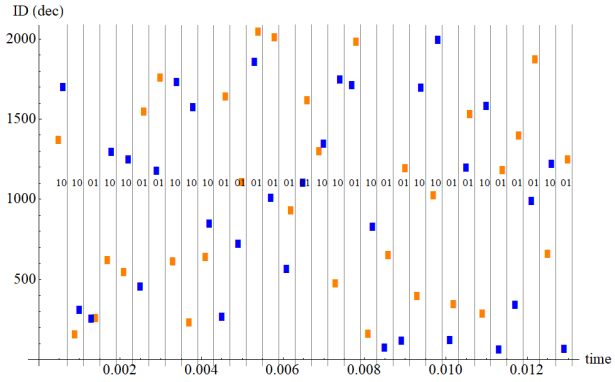

FIGURE 9. Minimax frame negotiation: frames on node A and node B (orange vs. blue).

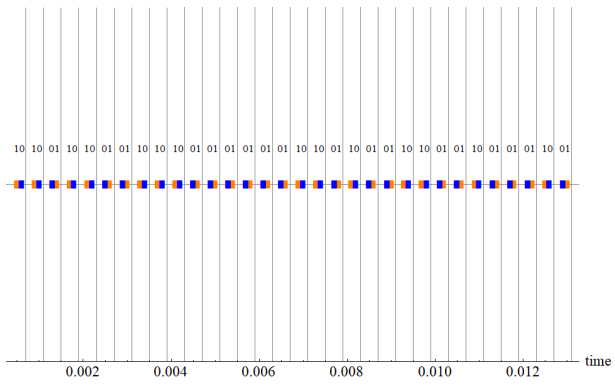

FIGURE 10. Minimax frame negotiation, frames arrived on the bus from Node A (orange) vs. Node B (blue).

ECU_A and ECU_B, generates two arrays of k random IDs $id_A = id_A^1, id_A^2, ..., id_A^k$, $id_B = id_B^1, id_B^2, ..., id_B^k$. These are sent on the bus at delay $i\Delta$. Due to the non-destructive arbitration of CAN the smaller ID will win the arbitration. We will call the transmitted IDs as $\mathsf{ID}_{\mathsf{min}}$ and $\mathsf{ID}_{\mathsf{max}}$.

Figure 8 shows the situations that may occur on the bus. There is a small probability that the IDs are identical on both nodes, this will be visible to an adversary since a single frame arrives on the bus (depicted in the figure as $\mathsf{ID}_{\mathsf{eq}}$). The probability for this is $\mathsf{Pr}_{bad} = 1/2^{11}$ in case of standard IDs and $\mathsf{Pr}_{bad} = 1/2^{29}$ in case of extended identifiers. In case of extended IDs the chances of this happening is 1 in 500 million, which is very low. Even if this is the case, the protocol will continue with the rest of the bits. While we can reduce the chance of a collision by adding random bytes in the data-field (our protocols work with empty data-fields), adding a data-field will also increase the transmission time and lead to a larger Δ (requiring a longer protocol run). This effect is undesired, thus we stay to the 0-length data-field. In the experiments, the value of Δ was first fixed to $200\mu s$. Then we endeavored to lower this value to $\Delta = 125\mu s$ which again proved to be sufficient (theoretically, at 500Kbit/s the time on the bus for a frame with no data-field is $106\mu s$, thus $212\mu s$ is the minimum cycle time for 2 Minimax frames). In this case (i) we set the bits to 00. Otherwise, the bits are set to 01 (i) or 10 (ii). To avoid confusions, we assume that the node that transmits the first $\mathsf{ID}_{\mathsf{max}}$ is node A. Assuming that no identical IDs occur, each frame carries exactly one bit of entropy and assuming no collision this leads $\mathsf{H}_{average}(k) \approx k$. Since two frames may occur each time on the bus, frames will be sent at 2Δ intervals and thus the protocol runtime is $\mathsf{T}(k) = 2k\Delta$.

The experiments that follow show data for the first 64 frames that arrive on the bus (32 for each of the two nodes that negotiate the key). In Figure 9 we show frames on each of the two nodes placed on the X-axis according to the arrival timestamp and on the Y-axis by the value of the ID in decimal. The IDs broadcasted by the nodes are random. There is no collision for the 2^{11} bit IDs, although this may happen and the protocol will cope with this. In Figure 10 we show frames as they arrived on the bus, according to the trace from CANoe, and the bits that have been exchanged. Figure 11 serves only as a summary by placing a check-mark over the frames that arrived at the expected time (this is verified by all frames). All frames are successfully checked.

Time-triggered Minimax Negotiation. Besides generating the IDs identical to the Minimax protocol, both nodes generate also an array of random bits which is to be kept secret, i.e., $b_A = b_A^1, b_A^2, ..., b_A^k, b_B = b_B^1, b_B^2, ..., b_B^k$. In the time-triggered Minimax key-exchange, frames with random IDs are sent by the nodes at intervals $(2i + b_i)\Delta$, i.e., either $2i\Delta$

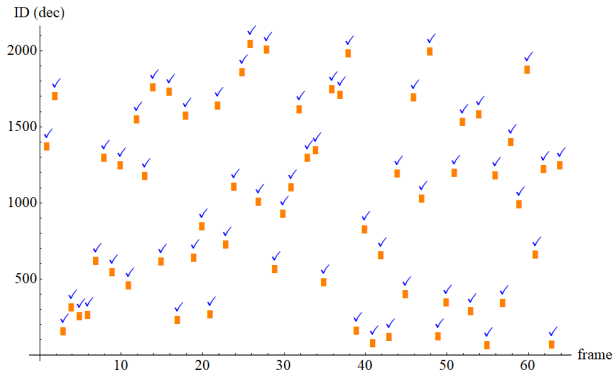

FIGURE 11. Minimax frame negotiation: expected arrival frame (as checkmark) vs. arrived frame (square).

or $(2i+1)\Delta$, according to the bit in the array. The case of distinct bits, $10, 01$ can be easily separated regardless of the value of the ID by the sender and receiver. Bits are visible to an adversary in the equality case $00, 11$ since in this case the frames will overlap and the inter-frame space will be smaller. Thus half of the bits are lost and $\mathsf{Pr}_{bad} = 1/2$ while the entropy carried by k frames is $\mathsf{H}_{average}(k) = k/2$. To avoid overloading the figure, we omit the 00 case which results in lost security bits anyway. The probability of ID colliding will require the same sending time, thus is it reduced to half when compared to the previous case. Statistically, this is included in the equality case $00, 11$ as it requires identical timings. The time increases since an additional Δ is required for the case when both frames are delayed, i.e., 11, leading to $\mathsf{T}(k) = 3k\Delta$. Figure 12 shows the situations that may occur on the bus.

The delay bit will be visible when the frames arrive one after another. This results in half of the bits being disclosed to an adversary. This apparent disadvantage opens door to an improvement by bootstraping the session key with EKE which will be discussed later. Since there is a possibility that both nodes send the frame at $(2i+1)\Delta$ we have to expand the periodicity at 3Δ which makes the runtime a bit longer.

We again provide experiments for the first 64 frames that arrive on the bus (32 for each of the two nodes). Figure 13 places frames on the X-axis according to the arrival timestamp and on the Y-axis by the value of the ID in decimal. We preserved the same random IDs for the frames as in the previous experiment but the periodicity now increases to 3Δ and thus the values on the X-axis of the plot are now distinct. To shorten the runtime of the protocol, in the experiments we forced Δ to $200\mu s$ which caused no problems. In Figure 14 we show frames as they arrived on the bus by using the trace from CANoe and the bits that have been exchanged (the delays vary according to the bits that are negotiated). Again, Figure 11 serves only as a summary on the expected arrival tine if the frames and all frames are successfully checked.

The entropy of the exchanged key can be further improved by harvesting additional bits based on the principle of the Minimax scheme. In case when 11 or 00 occur on the bus there will still be an ID that is smaller and thus wins the arbitration. Consequently, one additional bit can be harvested from the ID that wins and thus the entropy of the scheme can be set to $\mathsf{H}_{average}(k) = k$. However, our intention is to extract the key solely from the delay which further allows bootstrapping the key with EKE. For this reason we leave the scheme as it is and the improvement can be used when EKE is not a viable option.

Randomized time-triggered key exchange. In the randomized time-triggered key-exchange, nodes are selecting k random values in integer interval $[1..\ell]$, where $k < \ell$. Let $x_A^1, x_A^2, ..., x_A^k$ be the values for node A and $x_B^1, x_B^2, ..., x_B^k$ be the values for B. Frames with random identifiers are broadcast at time $x_A^i\Delta$ by node A and $x_B^i\Delta$ by node B. To extract the key, a 0 is added when a frame from A arrives and a 1 is added when a frame from B arrives on the bus.

In the experiments we used $\Delta = 200\mu s$ and $l = 512$. The bits of the key are extracted based on the delays between distinct frames from the same sender. In Figure 17 we show the arrival time for the first 32 frames and in Figure 18 for the last 32 frames. Due to randomized allocation of delays, collisions are expected and these are marked by an X (one collision occurred only in the second figure). Figure 19 gives an overview of the arrival time in conjunction with the ID of each frame (this is also randomly generated by each node).

The probability of frames to be sent at the same time (which discloses that delays are identical on both nodes and thus the two frames cannot originate from the same node) depends on ℓ. The probability that the ℓ selected delays from node A are all distinct from the delays selected by B is $(\ell - k)/\ell$ which means that the probability for a single collision and thus $\mathsf{Pr}_{bad} = 1 - (\ell - k)/\ell = k/\ell$. For simplicity, to estimate the entropy, we assume that no collision occurs (by setting the proper ℓ chances for collisions are small, in the experiments that we discuss next a single collision occurred). The adversary will observe $2k$ frames on the bus. To extract the key bits, since the delays between frames can be measured by the adversary as by genuine users, the adversary has to guess which frame was sent by which node. There are $(2k)!/k!$ possibilities for the frames of node A (node B will have the remaining k frames) and since the order of the k frames does not matter we have to divide this again by $k!$. This leads to entropy $\mathsf{H}_{average}(k) = -\log_2\left((k!)^2/(2k)!\right)$. The length of the interval ℓ can be accounted by taken the average number of frames that collide as k^2/ℓ and replacing k in the previous relation with the number of correct frames, i.e., $k - k^2/\ell$. Figure 20 gives an overview of the entropy extracted with the number of frames k and the length of the interval ℓ. Note that as ℓ increases the chance for a collision decreases (and the entropy increases) but so does the duration of the key-exchange protocol. Thus ℓ should be increased with care. The time to send all the ℓ frames is $\mathsf{T}(k) = \ell\Delta$.

Table 2 summarizes metrics for the four protocol versions. Next we improve on the last two protocols with a session-key bootstrap based on the EKE protocol.

All of the previous graphical depictions are based on real

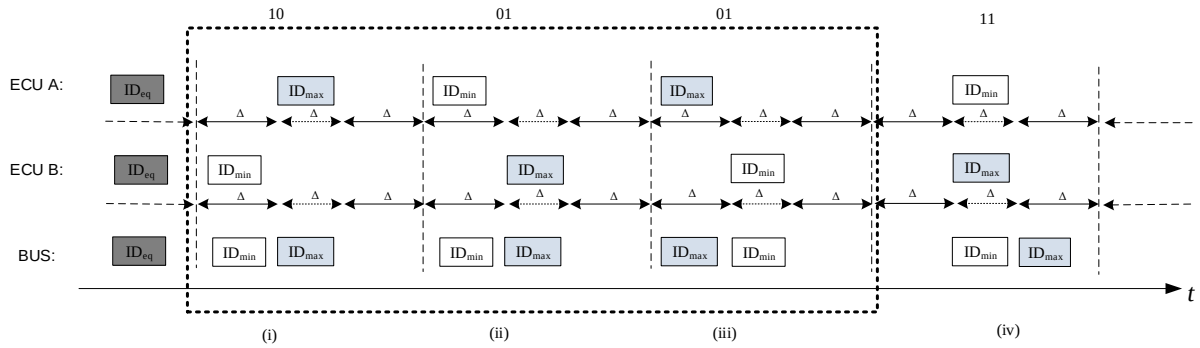

FIGURE 12. Time-triggered Minimax Key Exchange.

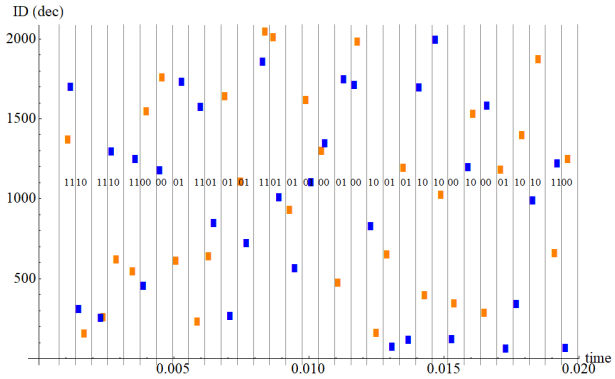

FIGURE 13. Time-triggered Minimax frame negotiation: frames on node A and node B (orange vs. blue).

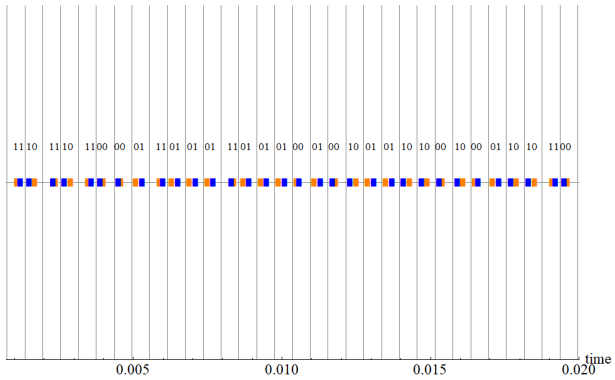

FIGURE 14. Time-triggered Minimax frame negotiation, frames arrived on the bus from Node A (orange) vs. Node B (blue).

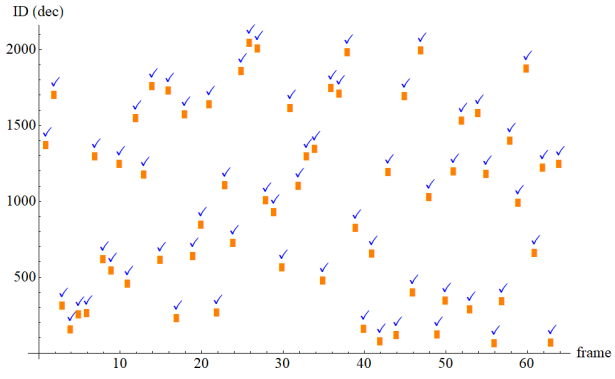

FIGURE 15. Time-triggered Minimax frame negotiation: expected arrival frame (as checkmark) vs. arrived frame (square).

experimental data. For a crisper image, we also illustrate some captures on the bus that were done with a Salae logic analyzer. Figure 21 shows the full 64 frames arriving on the bus during Time-triggered Minimax. Then in Figure 23 we illustrate the 01 case, i.e., the first frame arrives with no delay and the second is delayed. Similarly, Figures 22 and 24 illustrate the 11 case in slot 2 and 3 respectively, here both frames are sent with a delay. In Figure 25 we give a detailed plot for the 11 scenario.

IV. CRYPTOGRAPHICALLY RE-ENFORCED VERSIONS

We extend the last protocol versions with the Diffie-Hellman version of the Encrypted Key Exchange (EKE) protocol which allows bootstrapping an authentic key based on a low-entropy common secret. Finally we discuss a group extension that allows multiple nodes to negotiate a session key.

A. PIGGY-BACKED DIFFIE-HELLMAN EKE/SPEKE

While our proposal has the advantage of being fully implementable on software (which is not the case for the solution in [34]), it is less efficient in terms of bandwidth compared to [34] as we rely on one frame for exchanging a single bit. Here we turn this to our advantage by piggybacking frames with parts of a Diffie-Hellman [14] based keys shares of EKE [6]. This scheme achieves provable security against guessing attacks [5] and can be used to bootstrap a session key by using a small entropy shared secret. For a more compact representation however, we rely on elliptical-curves and thus we modify this scheme by using the main idea from the

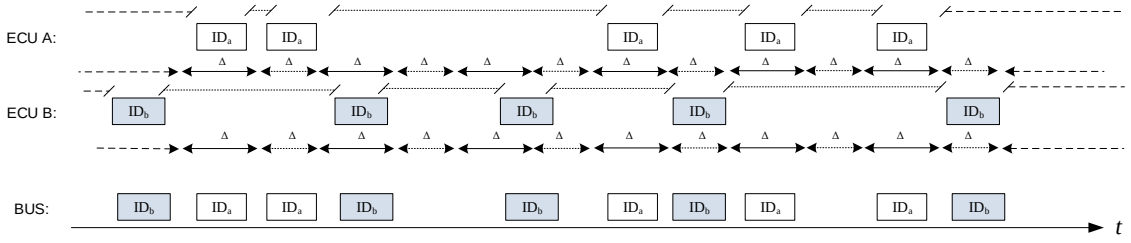

FIGURE 16. Randomized time-triggered key exchange.

TABLE 2. Comparison of the key-exchange schemes

	Data vs. Remote	Minimax	Time-triggered Minimax	Randomized Time-triggered
Pr_{bad}	$1/2$	$1/2^{11}$	$1/2$	k/ℓ
$\mathsf{H}_{average}$	$k/2$	k	$k/2$	$-\log_2\left(\frac{(k!)^2}{(2k)!}\right)$
T	$2k\Delta$	$2k\Delta$	$3k\Delta$	$\ell\Delta, \ell > k$

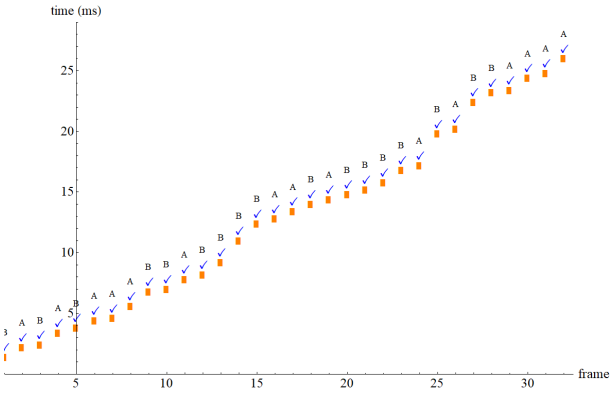

FIGURE 17. Arrival time for the first 32 frames with Randomized time-triggered key-exchange.

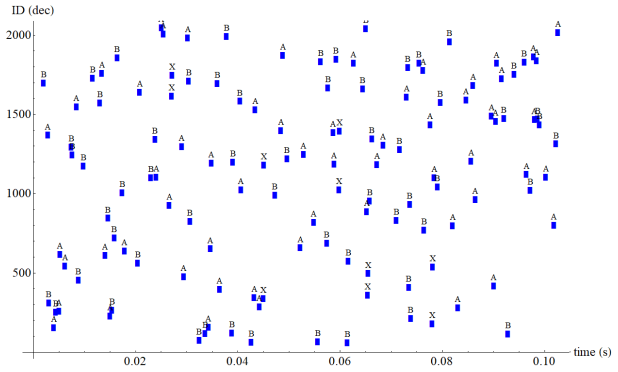

FIGURE 19. Frame arrival time and ID value with Randomized time-triggered key-exchange.

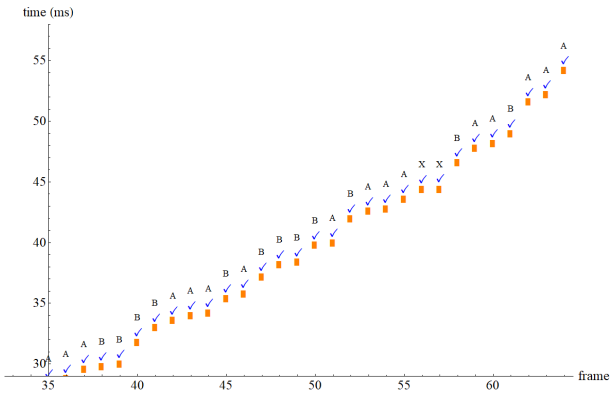

FIGURE 18. Arrival time for the first 32 frames with Randomized time-triggered key-exchange.

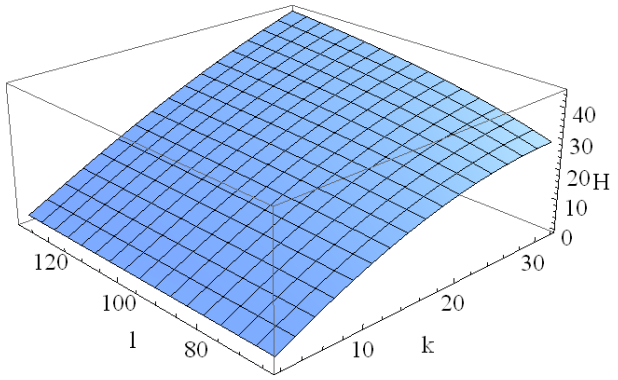

FIGURE 20. Entropy extracted from the randomized time-triggered key-exchange (approximation).

SPEKE protocol [22].

Both the time-triggered minimax and the randomized time-triggered key exchange are suitable for piggy-backing the EKE-DH frames. The minimax negotiation is not, since

FIGURE 21. All 64 pairs of frames arriving on the bus during Time-triggered Minimax (capture with Salae logic analyzer).

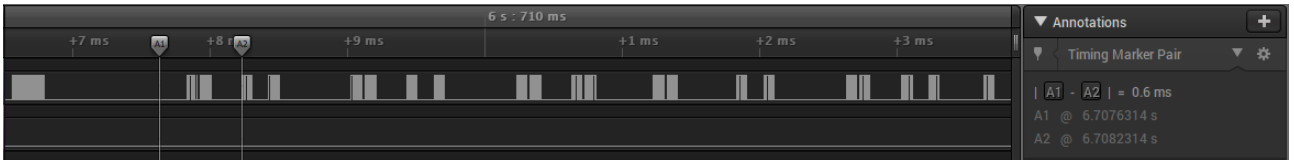

FIGURE 22. The 11 case (both frames with delays) of Time-triggered Minimax in slot 1 (between markers A2-A1).

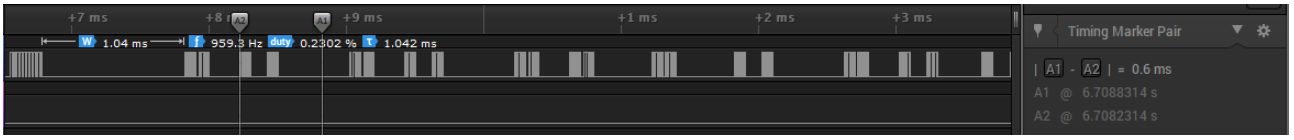

FIGURE 23. The 01 case (one frame with delay) of Time-triggered Minimax in slot 2 (between markers A2-A1).

FIGURE 24. Another 11 case (both frames with delays) of Time-triggered Minimax in slot 3 (between markers A2-A1).

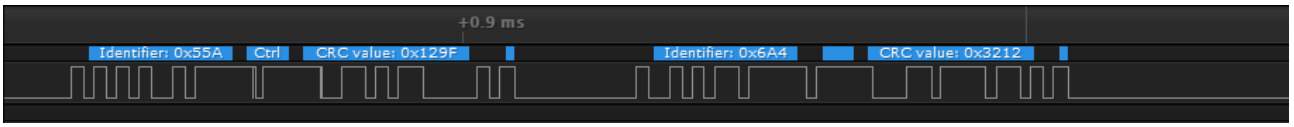

FIGURE 25. Detailed view of the 11 case (both frames with delays) of Time-triggered Minimax.

the nodes do not know in an a-priori manner the bits of the key that is going to be negotiated. For these schemes the EKE-DH can be used only after the key bits are exchanged. It is more efficient if the bits are piggy-backed to the frames and thus the last two protocol versions are more suitable for this kind of key negotiation. We illustrate this on the Time-triggered Minimax scheme which is easier to understand and implement.

Protocol 1 (Time-triggered Minimax with piggy-backed EKE/SPEKE). We assume that an elliptic curve $E/\mathbb{F}_q$ and fixed point $P = f(\widetilde{w}) \in E(\mathbb{F}_q)$ of prime order p as public system parameters. Here f is a function that maps the weak secret to a point P on the elliptical curve, similar to the choice of generator g in the SPEKE protocol [22]. The delay Δ is also fixed as system wide parameter (practical instantiations of these are to be discussed below). Each of the two nodes follows the procedures below:

1) $\mathsf{Setup}(1^k)$ in which ECU_A and ECU_B generate and keep secret two arrays of k random bits $b_A = b_A^1, b_A^2, ..., b_A^k$, $b_B = b_B^1, b_B^2, ..., b_B^k$, two arrays of k random IDs $id_A = id_A^1, id_A^2, ..., id_A^k$, $id_B = id_B^1, id_B^2, ..., id_B^k$, two integers $x_A \in \mathbb{Z}_p$ and $x_B \in \mathbb{Z}_p$ then compute $s_A = b_A x_A P = \{s_A^1, s_A^2, ..., s_A^k\} \in E(F_q)$, $s_B = b_B x_B P = \{s_B^1, s_B^2, ..., s_B^k\} \in E(F_q)$,
2) $\mathsf{SendCyclic}(i), i = 1..k$ in which $\mathsf{ECU}_\alpha, \alpha \in \{A, B\}$ at each clock tick $3i\Delta, i = 1..k$, if $b_\alpha^i = 1$ waits for more Δ ticks, then broadcasts a frame with identifier id_α^i and data-field containing s_α^i.

3) ExtractSessionKey(T) in which $\mathsf{ECU}_\alpha, \alpha \in \{A, B\}$, having the frames received on the bus along with their arrival timestamps $T = \{(id_1, t_1), (id_2, t_2), (id_3, t_3),..., (id_{2k}, t_{2k})\}$ checks if $|t_{2i-1} - t_{2i}| < \epsilon, i = 1..k$ and sets $b^i_\beta = \neg b^i_\alpha$ or otherwise sets $b^i_\beta = b^i_\alpha$, then recovers the shared Diffie-Hellman session key as $K_{ses} = x_\alpha b_\alpha^{-1} Q$ (where Q is the point recovered from the k packets).
4) ConfirmSessionKey(K_{ses}) at which having recovered the session key ECU_A places on the bus a frame containing $MAC_{K_{ses}}(s_B)$ and ECU_B places on the bus a frame containing $MAC_{K_{ses}}(s_A)$ (we assume that the cryptographic MAC is truncated to 64 bits such that it fits a single CAN frame, less may be in order). If the session key is confirmed by both parties, then the weak secret $\widetilde{w}$ is updated to $\widetilde{w} = MAC_{\widetilde{w}}(K_{ses})$.
5) Abort(T) is the action by which $\mathsf{ECU}_\alpha, \alpha \in \{A, B\}$ aborts the protocol and signals this by error frames if any of the following is met: i) more than $2k$ frames arrived on the bus, i.e., $|T| > 2k$, ii) the frames in T are not a pair-wise succession id_α, id_β with $\alpha \neq \beta$ where one of the IDs is known by the the sender node or iii) $|t_{2i-1} - t_{2i}| > 3\Delta$.

The protocol ends successfully if the cryptographic MAC can be verified by both nodes.

A practical instantiation of this scheme that we use in our experiments consists in setting $k = 21$, thus 42 frames are sent in total. The elliptical curve is defined over a 160-bit field, thus 20 bytes are needed to send the X-coordinate of each point. One additional bit is needed for the sign of the point, that is, we use the compressed form of the points on the curve.

B. SECURITY ARGUMENT

We now give arguments on the security of the protocol. Since SPEKE is part of existing standards, i.e., IEEE 1363.2 [16] and ISO 1170-4 [21], and has been also analyzed by the community [43], [40] and [19] a formal security analysis would be out-of-scope for the current work (we are concerned with deploying cryptographic building blocks on an automotive-grade controller/bus). We did consider the use of a model-checker like AVISPA [1] for verifying security, however, we noticed that in the AVISPA library of protocols, the existing model for SPEKE outputs no attack http://www.avispa-project.org/library/SPEKE.html. Recently however, attacks and fixes were reported on the SPEKE protocol by [19]. This suggests that the model-checker cannot cope with the mathematical details behind SPEKE. At best, what we could do is to build model that accounts for the recent vulnerabilities from [19]. These attacks (an impersonation attack in parallel sessions that is inapplicable in case of our work and a key malleability attack which does not leak the shared secret) are however easy to fix as shown in [19]. Thus, in what follows, we discuss the rationale behind the protocol design and we are confident on existing analysis and fixes proposed by the community, e.g., [43], [40] and [19].

Our scheme continuously increases the entropy of the weak secret $\widetilde{w}$ by repeatedly ratcheting the newly exchanged secret session on the previously know weak secret $\widetilde{w}$. Deriving point $P = f(\widetilde{w})$ from the weak shared secret $\widetilde{w}$ prohibits an adversary from playing as man-in-the-middle since the adversary cannot compute the point P (this is the concept behind the SPEKE protocol [22]). Consequently, if an adversary disconnects one of the genuine nodes from the bus and sends his own piggy-backed frames this will carry some value $b_{adv} x_{adv} P'$ where $P' \neq P$. The resulting key-share will be $x_\alpha x_{adv} P'$ for the genuine node α and $x_{adv} x_\alpha P$ which will not match. The security will held also in front of probing adversaries, e.g., [24], assuming that they have no access to the weak secret $\widetilde{w}$. The entropy increases in time with every bit that such an adversary is unable to extract.

If the weak shared secret $\widetilde{w}$ is unavailable then the protocol achieves security only in front of a weaker form of adversary. Namely, the protocol is still secure only if we assume that the adversary is unable to unplug nodes from the bus nor is he able to probe to bus in order to learn the bits $b^i_\alpha, \alpha \in \{A, B\}, i = 1..k$. Otherwise, if the adversary can disconnect the genuine node from the bus, then he can successfully impersonate it (this is in fact the case in the original proposal from [34]). We believe that such a scenario (when nodes can be unplugged from the bus) is not so easy to set on an in-vehicle network and even if this is the case then the existence of a weak secret of some sort $\widetilde{w}$ should be realistic for ECUs that share a common history. As stated in the introductory section, by following the resurrecting-duckling paradigm [38], components inside the car may be imprinted with a secret derived from the CAN data recorded when the car is started for the first time or imprinted with a factory secret key. While both these alternatives lead to a weak shared secret (e.g., some similarities may exist between distinct cars from the same manufacturer), the guessing resilient EKE/SPEKE protocol will help to bootstrap a cryptographically strong shared secret-key.

C. IMPLEMENTATION DETAILS

In the piggybacked encrypted key exchange protocol, the generated random bits are used to calculate the ECC public key for each node. The public key is exchanged on the CAN bus by each node using 42 frames of one byte (20 frames for the public key data bytes and 1 frame for the sign of the point, done by each node using one of the last two key-exchange protocols). We can extend the size of the data-field to more than a single byte and reduce the number of frames. However, we want a sufficient number of bits for masking in the key exchange and thus we consider that 21 frames of one byte is the better option. After all the messages are received, the random bits of the other node are extracted based on the populated timestamp arrays. Using the random bits and the public key, each node calculates its session key. For verification of having the same session key, each node has sent on the CAN bus a MAC value of the received public key, using the session key. The received MAC value is compared

with a computed MAC value, based on its own public key, using the session key.

For the encrypted key exchange mechanisms, we have used MIRACL (Multiprecision Integer and Rational Arithmetic Cryptographic) [33] which is a C/C++ software library covering a wide range of applications including elliptic curve cryptography (ECC).

For exchanging the point on the curve we used point compression and thus exchanging only the X coordinate of a point. The curve that we used was 160-bit and thus 20 bytes plus 1 for the sign of the Y coordinate is sufficient. This results in the 21 bytes that are exchanged for key extraction by the EKE protocol.

Table 3 gives a summary of the parameters and runtime in all the experiments so far. A key can be exchanged in roughly a few dozens milliseconds. The Minimax is more efficient in terms of extracted entropy since each frame contributes to the entropy of the key in our experiments we have also lowered the delay to $125\mu s$ and thus 64 bits could be exchanged in $16ms$. The size of the extracted keys increases linearly with time, so Table 3 is enough to infer the time for extracting keys of any size. In case of the EKE-DH we consider that the size of the entropy of the shared key is the entropy of the exchanged Diffie-Hellman key, i.e., 160 bits as we worked on an 160-bit curve. In this case, the time represents the time to send the 42 frames on the bus, i.e., $12ms$, to which the computational time compute the shares and extract the key has to be added, i.e., $98ms$ if SPEKE alone is used. We now discuss more on computational requirements for the piggy-backed versions. Table 4 summarizes the computational time for the EKE and SPEKE protocols and contrasts these to the regular Diffie-Hellman on an NIST 192-bit curve (which was available in the same cryptographic library).

Computing the piggy-backed key will require less than $100ms$ depending on the platform and implementation. To check that a point belongs to the curve, we use the Boolean returned by the *set* function available in MIRACL for setting points based on the X coordinate and their sign. A new X-coordinate and sign are generated from the value $\widetilde{w}$ by seeding it to the PRNG (pseudo-random number generator) available in the MIRACL library. If the resulting point does not belong to the curve, a new output is retrieved from the PRNG until the point belongs to the curve. On average, two attempts are needed since there are 50% chances that the resulting y^2 from replacing x in the equation of the curve is indeed a square (this was verified experimentally). We benchmarked the code and this operation (point selection of the curve) increases to computational time to around 25%, i.e., it requires less than $10ms$ compared to computing the regular Diffie-Hellman key-share (which requires the generation of a random scalar a and one point multiplication aP).

The regular Diffie-Hellman is 4–6 times faster due to specific optimizations which we could enable on the C implementation. For the moment we were not able to enable such optimizations on the C++ implementations that we did for EKE and SPEKE. Since EKE requires one additional point multiplication and SPEKE requires deriving a new point P on the curve, these should be roughly two times slower than the regular Diffie-Hellman. With specific optimizations this should be the expected computational time. A more interesting option is to use fast curves such as Curve25519 [7] or FourQ [12]. Curve25519 uses a 255-bit modulus that is larger than the 160-bit modulus which was chosen by us for the compact representation (bandwidth is a major concern for in-vehicle networks while CAN packets are limited to 64 bits). For completeness however, we also evaluated the FourQ implementation[1] which according to [12] is faster than Curve25519 and we compared it to our SPEKE implementation in Linux on a standard Core i7 PC. For extracting the Diffie-Hellman key share, i.e., computing abP from a and bP, the FourQ code was almost 4 times faster than the 160-bit curve from MIRACL. Porting this on Infineon will require more work but the results suggest that, with specific optimizations and a faster curve, a key can be exchanged in a dozen milliseconds or less.

Both EKE and SPEKE are affordable and they allow the secure exchange of a secret key based on a low-entropy secret. If both the masking of the share with the random bits is done, by computing xP, and extraction of P according to the SPEKE specifications then around $80ms$ should be required for both computing the key shares and extracting the session key. This requirement is modest and should be affordable for most modern automotive-grade controllers assuming that a key-exchange session is not done very often, e.g., only when the nodes re-join the bus.

D. MULTI-PARTY VERSION OF THE SCHEME

The proposed schemes can be immediately extended to session key negotiation between more than a single pair of nodes. Various ways to achieve group extensions of the Diffie-Hellman protocol have been previously discussed in works such as [39] or [25] and these can be applied to the current proposals.

We outline here a protocol version that is close to the one in [39]. The CAN nodes $\mathsf{ECU}_i, i = 1..n$ proceed in pairs. First, ECU_1 and ECU_2 exchange a session key using the TT-Minimax with EKE-DH. Let this session key be x_1x_2P. Then, based on the session key, they generate by using a key derivation function KD the seed for a PRNG that generates a new sequence of bits b_{12} and a new random value x_{12} (since these values are generated by PRNGs with identical seed, the values will be identical on the side of ECU_1 and ECU_2). These values are to be used in negotiating a key with ECU_3 and so on. Finally, for negotiation with ECU_n, all previous units $\mathsf{ECU}_1, \mathsf{ECU}_2, ..., \mathsf{ECU}_{n-1}$ have agree with a key that is $x_{123...n-2}x_{n-1}P$ and this key is used to share the key with ECU_n. The resulting key will be $x_{123...n-1}x_nP$. Figure 26 gives an overview of the protocol actions. The squares represent the ECUs and the gray ovals the key that is exchanged between two ECUs or a logical entity that groups

[1] https://github.com/Microsoft/FourQlib

TABLE 3. Summary of experimental parameters and results

	Data vs. Remote	Minimax	Time-triggered Minimax	Randomized Time-triggered	SPEKE TT-Minimax
Δ (μs)	200	125	200	200	200
$H_{average}$ (bits)	32	64	32	124	160
T (ms)	25	16	38	102	12 (bus) + 98 (comp.)

TABLE 4. Computational time for the cryptographic blocks on Infineon TC 297

	Diffie-Hellman using NISTP192	EKE (Diffie-Hellman based)	SPEKE
Share (compute aP)	12.1 ms	74.32 ms	53.61 ms
Recover (extract abP)	13.93 ms	73.84 ms	44.39 ms

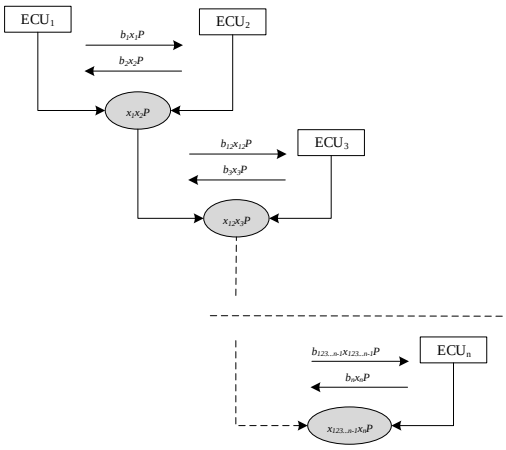

FIGURE 26. Suggested group keying with EKE-DH.

all previous ECUs under the same key. The logical entity will be instantiated by the last of the ECUs that is grouped under it. Ideally, all nodes behind a logical entity can broadcast the frame since the content is identical and a single frame will appear on the bus. This may however trigger errors in case that synchronization is not perfect between nodes.

Electronic control units with low computational power, that cannot handle public key encryption, proceed in identical fashion to what is suggested in Figure 26 but they simply skip the EKE-DH negotiation and the session key follows from the frame timing alone. That is, all the previous schemes apply in a cascade fashion from ECU_1 to ECU_n.

V. CONCLUSIONS

Secure key-exchange on the CAN bus is a relevant topic since the large majority of existing schemes for assuring authenticity on CAN require a secret shared key. Our work provides a protocol suite for securely exchanging session keys over the CAN bus which can be fully implemented on the software layer. The basic versions of the schemes do not require cryptographic capabilities from the CAN nodes. Still, nodes that can handle Elliptical Curve Cryptography can extract larger session keys in an authenticated manner by using the modified EKE/SPEKE versions of the Diffie-Hellman protocol. Convincing experimental results are provided on high-end controllers with Infineon Aurix cores, i.e., TC297, TC277, both for the simple bus-based key negotiation as well as for the crypto-based EKE/SPEKE-DH key sharing. We discuss both two-party versions of the scheme as well as multi-party versions and show that these scale well with the number of nodes.

REFERENCES

[1] A. Armando, D. Basin, Y. Boichut, Y. Chevalier, L. Compagna, J. Cuéllar, et al. The avispa tool for the automated validation of internet security protocols and applications. In International conference on computer aided verification, pages 281–285. Springer, 2005.

[2] AUTOSAR. Specification of Crypto Abstraction Library, 4.2.2 edition, 2015.

[3] AUTOSAR. Specification of Secure Onboard Communication, 4.3.1 edition, 2017.

[4] G. Bella, P. Biondi, G. Costantino, and I. Matteucci. Toucan: A protocol to secure controller area network. In Proceedings of the ACM Workshop on Automotive Cybersecurity, pages 3–8. ACM, 2019.

[5] M. Bellare, D. Pointcheval, and P. Rogaway. Authenticated key exchange secure against dictionary attacks. In International conference on the theory and applications of cryptographic techniques, pages 139–155. Springer, 2000.

[6] S. M. Bellovin and M. Merritt. Encrypted key exchange: Password-based protocols secure against dictionary attacks. In Research in Security and Privacy, 1992. Proceedings., 1992 IEEE Computer Society Symposium on, pages 72–84. IEEE, 1992.

[7] D. J. Bernstein. Curve25519: new Diffie-Hellman speed records. In International Workshop on Public Key Cryptography, pages 207–228. Springer, 2006.

[8] S. Checkoway, D. McCoy, B. Kantor, D. Anderson, H. Shacham, S. Savage, K. Koscher, A. Czeskis, F. Roesner, T. Kohno, et al. Comprehensive experimental analyses of automotive attack surfaces. In USENIX Security Symposium. San Francisco, 2011.

[9] K.-T. Cho and K. G. Shin. Fingerprinting electronic control units for vehicle intrusion detection. In 25th USENIX Security Symposium, 2016.

[10] K.-T. Cho and K. G. Shin. Viden: Attacker identification on in-vehicle networks. In Proceedings of the 2017 ACM SIGSAC Conference on Computer and Communications Security, pages 1109–1123. ACM, 2017.

[11] 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 Transactions on Information Forensics and Security, 2018.

[12] C. Costello and P. Longa. FourQ: Four-dimensional decompositions on a Q-curve over the mersenne prime. In Advances in Cryptology - ASIACRYPT 2015 - 21st Intl. Conf. on the Theory and Application of Cryptology and Information Security, pages 214–235, 2015.

[13] L. Dariz, M. Selvatici, M. Ruggeri, G. Costantino, and F. Martinelli. Trade-off analysis of safety and security in can bus communication. In 2017 5th

IEEE International Conference on Models and Technologies for Intelligent Transportation Systems (MT-ITS), pages 226–231. IEEE, 2017.
[14] W. Diffie and M. Hellman. New directions in cryptography. IEEE transactions on Information Theory, 22(6):644–654, 1976.
[15] D. Dolev and A. Yao. On the security of public key protocols. IEEE Transactions on information theory, 29(2):198–208, 1983.
[16] I. P. W. Group et al. P1363. 2: Standard specifications for password-based public-key cryptographic techniques. Draft available at: http://grouper. ieee. org/groups/1363.
[17] B. Groza, S. Murvay, A. V. Herrewege, and I. Verbauwhede. Libra-can: Lightweight broadcast authentication for controller area networks. ACM Transactions on Embedded Computing Systems (TECS), 16(3):90, 2017.
[18] B. Groza, L. Popa, and S. Murvay. INCANTA - intrusion detection in controller area networks with time-covert cryptographic authentication. In International Workshop on Cyber Security for Intelligent Transportation Systems (ESORICS'18 Workshops), 2018.
[19] F. Hao and S. F. Shahandashti. The SPEKE protocol revisited. In International Conference on Research in Security Standardisation, pages 26–38. Springer, 2014.
[20] O. Hartkopp, C. Reuber, and R. Schilling. MaCAN-message authenticated CAN. In 10th Int. Conf. on Embedded Security in Cars (ESCAR 2012), 2012.
[21] I. S. o. I. T. ISO. Security techniques, key management, part 4: "mechanisms based on week secrets," iso/iec 11770-4. 2006.
[22] D. P. Jablon. Extended password key exchange protocols immune to dictionary attack. In Proceedings of IEEE 6th Workshop on Enabling Technologies: Infrastructure for Collaborative Enterprises, pages 248–255. IEEE, 1997.
[23] S. Jain and J. Guajardo. Physical layer group key agreement for automotive controller area networks. In Conference on Cryptographic Hardware and Embedded Systems, 2016.
[24] S. Jain, Q. Wang, M. T. Arafin, and J. Guajardo. Probing attacks on physical layer key agreement for automotive controller area networks (extended version). arXiv preprint arXiv:1810.07305, 2018.
[25] Y. Kim, A. Perrig, and G. Tsudik. Group key agreement efficient in communication. IEEE transactions on computers, 53(7):905–921, 2004.
[26] M. Kneib and C. Huth. Scission: Signal characteristic-based sender identification and intrusion detection in automotive networks. In Proceedings of the 2018 ACM SIGSAC Conference on Computer and Communications Security, pages 787–800. ACM, 2018.
[27] R. Kurachi, Y. Matsubara, H. Takada, N. Adachi, Y. Miyashita, and S. Horihata. CaCAN - centralized authentication system in CAN (controller area network). In 14th Int. Conf. on Embedded Security in Cars (ESCAR 2014), 2014.
[28] C. Lin, Q. Zhu, C. Phung, and A. Sangiovanni-Vincentelli. Security-aware mapping for can-based real-time distributed automotive systems. In 2013 IEEE/ACM International Conference on Computer-Aided Design (ICCAD), pages 115–121, Nov 2013.
[29] C.-W. Lin, Q. Zhu, and A. Sangiovanni-Vincentelli. Security-aware modeling and efficient mapping for CAN-based real-time distributed automotive systems. IEEE Embedded Systems Letters, 7(1):11–14, 2015.
[30] Y. Liu, H.-H. Chen, and L. Wang. Physical layer security for next generation wireless networks: Theories, technologies, and challenges. IEEE Communications Surveys & Tutorials, 19(1):347–376, 2017.
[31] C. Miller and C. Valasek. Adventures in automotive networks and control units. DEF CON, 21:260–264, 2013.
[32] C. Miller and C. Valasek. Remote exploitation of an unaltered passenger vehicle. Black Hat USA, 2015, 2015.
[33] MIRACL Ltd. Multiprecision Integer and Rational Arithmetic C Library – the MIRACL Crypto SDK. https://github.com/miracl/MIRACL. Accessed: 2018-12-11.
[34] A. Mueller and T. Lothspeich. Plug-and-secure communication for can. CAN Newsletter, pages 10–14, 2015.
[35] P.-S. Murvay and B. Groza. Source identification using signal characteristics in controller area networks. IEEE Signal Processing Letters, 21(4):395–399, 2014.
[36] R. L. Rivest, A. Shamir, and L. Adleman. A method for obtaining digital signatures and public-key cryptosystems. Communications of the ACM, 21(2):120–126, 1978.
[37] S. U. Sagong, X. Ying, A. Clark, L. Bushnell, and R. Poovendran. Cloaking the clock: emulating clock skew in controller area networks. In Proceedings of the 9th ACM/IEEE International Conference on Cyber-Physical Systems, pages 32–42. IEEE Press, 2018.
[38] F. Stajano and R. Anderson. The resurrecting duckling: Security issues for ad-hoc wireless networks. In International workshop on security protocols, pages 172–182. Springer, 1999.
[39] D. G. Steer, L. Strawczynski, W. Diffie, and M. Wiener. A secure audio teleconference system. In Proceedings on Advances in Cryptology, CRYPTO '88, pages 520–528, Berlin, Heidelberg, 1990. Springer-Verlag.
[40] Q. Tang and C. J. Mitchell. On the security of some password-based key agreement schemes. In International Conference on Computational and Information Science, pages 149–154. Springer, 2005.
[41] S. Woo, H. J. Jo, I. S. Kim, and D. H. Lee. A Practical Security Architecture for In-Vehicle CAN-FD. IEEE Trans. Intell. Transp. Syst., 17(8):2248–2261, Aug 2016.
[42] Y. Xie, G. Zeng, R. Kurachi, H. Takada, and G. Xie. Security/timing-aware design space exploration of can fd for automotive cyber-physical systems. IEEE Transactions on Industrial Informatics, 15(2):1094–1104, 2019.
[43] M. Zhang. Analysis of the speke password-authenticated key exchange protocol. IEEE Communications Letters, 8(1):63–65, 2004.

BOGDAN GROZA is Professor at Politehnica University of Timisoara (UPT). He received his Dipl.Ing. and Ph.D. degree from UPT in 2004 and 2008 respectively. In 2016 he successfully defended his habilitation thesis having as core subject the design of cryptographic security for automotive embedded devices and networks. He has been actively involved inside UPT with the development of laboratories by Continental Automotive and Vector Informatik. Besides regular participation in national and international research projects in information security, he lead the CSEAMAN project (2015-2017) and currently leads the PRESENCE project (2018-2020), two national research projects dedicated to automotive security.

LUCIAN POPA started his PhD studies in 2018 at Politehnica University of Timisoara (UPT). He graduated his B.Sc in 2015 and his M.Sc studies in 2017 at the same university. He has a background of 4 years as a software developer and later system engineer in the automotive industry as former employee of Autoliv (2014 - 2018) and current employee of Veoneer (2018 - present). His research interests are in automotive security with focus on the security of in-vehicle buses.

PAL-STEFAN MURVAY is Lecturer at Politehnica University of Timisoara (UPT). He graduated his B.Sc and M.Sc studies in 2008 and 2010 respectively and received his Ph.D. degree in 2014, all from UPT. He has a 10-year background as a software developer in the automotive industry. He worked as a postdoctoral researcher in the CSEAMAN project and is currently a senior researcher in the PRESENCE project. He also leads the SEVEN project related to automotive and industrial systems security. His current research interests are in the area of automotive security.