CANTO - Covert AutheNtication with Timing channels over Optimized traffic flows for CAN

Bogdan Groza, Lucian Popa and Pal-Stefan Murvay

***Abstract*—Previous research works have endorsed the use of delays and clock skews for detecting intrusions or fingerprinting controllers that communicate on the CAN bus. Recently, timing characteristics of CAN frames have been also used for establishing a covert channel for cryptographic authentication, in this way cleverly removing the need for cryptographic material inside the short payload of data frames. However, the main drawback of this approach is the limited security level that can be achieved over existing CAN bus traffic. In this work we significantly improve on this by relying on optimization algorithms for scheduling CAN frames and deploy the covert channel on optimized CAN traffic. Under practical bus allocations, we are able to extract 3-5 bits of authentication data from each frame which leads to an efficient intrusion detection and authentication mechanism. By accumulating covert channel data over several consecutive frames, we can achieve higher security levels that are in line with current real-world demands. To prove the correctness of our approach, we present experiments on automotive-grade controllers, i.e., Infineon Aurix, and bus measurements with the use of industry standard tools, i.e., CANoe.**

***Index Terms*—Authentication, CAN bus, Covert channel**

I. Introduction and motivation

Following small demonstrations of attacks on vehicle buses inside simulation environments, which can be traced as early as the work in [1], the feasibility of attacking real-world vehicles has been decisively proved by recent works such as [2], [3], [4], etc. Most of the security problems of in-vehicle networks stem from the fact that the Controller Area Network (CAN) has no intrinsic security mechanisms. The CAN bus is able to transmit up to 8 bytes of data in a single frame at a maximum bit rate of 1Mbit/s. The structure of the CAN frame is depicted in Figure 1. To avoid collisions, arbitration is performed based on the ID field which uses the first 11 bits of the frame (or 29 bits if extended identifiers are used), while a 15 bit CRC (Cyclic Redundancy Check) aids in the identification of transmission errors. CAN also uses bit stuffing to avoid synchronization loss in long sequences with bits of the same value, i.e., after five consecutive identical bits, an additional bit of opposite value is introduced to force a transition.

The research community has answered with dozens of proposals for securing the CAN bus. As expected, most of these rely on the use of cryptographic Message Authentication Codes (MACs) (e.g., [5], [6], [7], [8], or more recently [9] and many others). But due to the limited size of the CAN frame,

Bogdan Groza, Lucian Popa and Pal-Stefan Murvay are with the Faculty of Automatics and Computers, Politehnica University of Timisoara, Romania, Email: bogdan.groza@aut.upt.ro, lucian.popa.lp@gmail.com, pal-stefan.murvay@aut.upt.ro

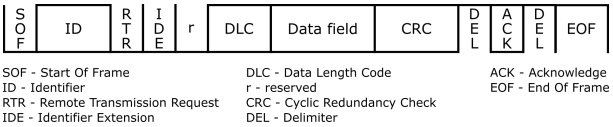

Fig. 1. Format of CAN data frames

i.e., 64 bits, two options have been commonly discussed in the literature: truncating the MACs so that they fit inside the frame along with legitimate data or sending the MACs as a distinct packet. The last option creates additional problems since sending a new authentication frame for each regular frame increases the bus-load and does not cope with real-time demands as we later discuss in our analysis. The first procedure, MAC truncation, is supported by the more recent security specifications introduced in the AUTOSAR [10] architecture which require 24 bits of security for a CAN frame. However, reserving 24 bits out of the 64 bit CAN frame payload for security may not be convenient as this represents 37% of the payload. Additionally, the standard specifies 8 bits for a freshness parameter, leading to 32 bits reserved for security purposes and thus 50% of the frame becomes unusable for regular data. In general, it seems that including cryptographic material in CAN frames remains somewhat problematic as the small packet size of CAN is hardly able to cope with the required level of security. A third option is to hide the authentication bits by using alternative physical layers such as CAN+ [11] which is an extension of CAN. Such an approach was proposed in [12]. However, CAN+ transceivers do not exist inside vehicles and due to the migration to CAN-FD it seems unlikely for CAN+ to be adopted by the automotive industry.

Context and motivation. In a recent work, we exploited the fine-grained control of timer-counter circuits in a constructive manner by designing a covert timing channel for cryptographic authentication [13]. This proposal has the merit of allowing authentication data to be carried outside the limited 64 bit payload of the CAN frame. However, the problem with the work in [13] is that performance degrades significantly when the covert channel is placed over existing CAN-bus traffic. A similar approach for creating a covert authentication channel based on frame arrival time can be found in [14], but the achieved security level is very limited at 1 covert bit for each CAN frame. Moreover, the authentication in [14] is dedicated to the transmitter alone, not to the content of the frame. By optimizing CAN traffic, our approach can obtain much higher data-rates on the covert channel. Of course, in case

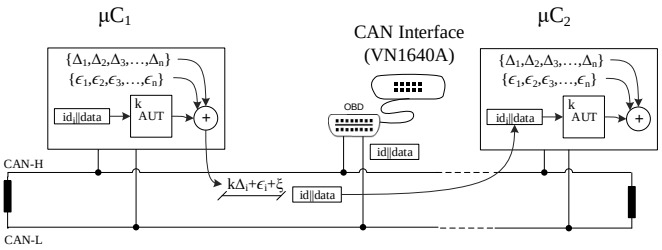

Fig. 2. Basic depiction of the addressed scenario: ECUs sending/receiving packets on the CAN bus, authentication data is encoded in delays

of multiple senders, this would require time synchronization between senders. Traditional implementations of CAN assume an asynchronous behavior of nodes and the absence of a common time base. However, recent approaches from the industry provide support for time-synchronization on all three major in-vehicle buses Ethernet, CAN and FlexRay, according to AUTOSAR standard specifications from [15]. The recent AUTOSAR specifications for time synchronization on CAN, give a caveat on the worst-case accuracy for time masters, gateways and even for slave nodes, by placing it at $10\mu s$, i.e., section 4.1 in [16]. This shows that the requirements of our approach are realistic and in-line with current trends from the industry. Time synchronization is also a requirement in case of time-triggered communication on TT-CAN (Time-triggered CAN) and thus present in real-world applications.

Design intentions. We create a covert authentication channel, which leaves the bits of the CAN frames unchanged, and increases the data rate of the channel by relying on efficient frame scheduling. The addressed scenario is briefly outlined in Figure 2. To avoid overloading the figure, only two ECUs (Electronic Control Units) and one external device (potentially a CANcase) are depicted, but there are no restrictions regarding the number of ECUs or external devices in our scenario. The ECUs communicate messages in the usual way at predefined periods $\Delta_i, i = 1..n$ for each of the IDs defined on the bus (we assume n such IDs), to which a small drift ϵ_i and a small authentication delay ξ are added (more details in the forthcoming sections). The main advantages of a covert authentication channel on the CAN bus are the following:

- it does not consume bits from the frame data-field which is limited at 64 bits,
- it covertly embeds authentication data in the frame that carries the data, without requiring an additional authentication frame,
- it does not increase the bus-load since authentication data is hidden in delays.

The covert channel that we create for message authentication on CAN has to be perceived as complementing rather than competing with regular MAC-based solutions. It is obvious that the use of regular MACs is more efficient, but it is also undeniable that the limited size of the data-field (64 bits) makes the inclusion of such MACs difficult and MAC truncation is the only option which in turn significantly cuts on the security level (adding additional authentication frames is likely impractical as we later discuss). In contrast, the covert authentication channel can bring an additional layer of protection on existing traffic without penalties. In fact, the optimized traffic allocations that we discuss directly improve on the worst case arrival time for genuine frames since there are fewer or hopefully no frame collisions on the bus. A comparison between the capacity of the covert-channel and that of the regular, intended communication channel, would be biased since the former builds on minuscule physical fluctuations of the later (not surprisingly, covert channels do have a smaller data rate).

The main concept behind how frame authentication works in our proposal, i.e., encoding authentication data in delays and adding optimizations for frame timings, is detailed in Figure 3. CAN frames, depicted by the identifier field ID, arrive on the bus in a cyclic manner (to avoid overloading the figure, we omit the data-field, but this is used in the message authentication code along with the ID). While on-event frames may also exist on the CAN bus, the majority of the CAN traffic is cyclic in nature and we focus our work on authenticating such traffic. We depict identifiers for 3 distinct delays $\Delta = 10, 20, 50ms$. A drift ξ is added to each delay which carries authentication data in a covert manner. In principle ξ is the last byte of a cryptographic message authentication code (MAC). This MAC is computed over the content of the entire frame and will be distinct for every frame assuming proper use of freshness parameters, e.g., timestamps or counters. To avoid overloading the figure we omit such details in the graphical outline. Due to improper allocation of CAN traffic, several packets may need to be transmitted at the same time (this is suggested by packets highlighted in gray). Such overlaps may not be a problem from a transmission point of view, however, they affect the expected arrival time and thus the data-rate of the covert channel. To avoid such situations, we use an additional delay $\epsilon_i, i = 1..3$ in order to allocate traffic in an optimal manner and keep frame inter-distance at a maximum. Figure 4 gives a crisper image on why poorly optimized traffic is problematic for a covert timing channel. The left side of the figure shows the inter-transmission times between frames as recorded in a real-world vehicle. While the entire traffic is cyclic, the inter-transmission time is noisy and deviations from the expected arrival are common. To improve performance we further rely on optimization algorithms. The right side of Figure 4 shows inter-transmission times after the traffic is optimized. The same bus-load and the same number of IDs is used, but the inter-transmission time now follows a clearer pattern. As expected and later proved by the experiments, the covert channel will have a superior bitrate over the optimized traffic.

A. Related work

Covert timing channels have been well explored in computer networks, e.g., [17], [18], [19], but, except for the aforementioned recent papers [13] and [14], we are unaware of the use of such channels for securing in-vehicle communication.

Table I attempts for a comparative summary between the two existing proposals [13], [14] for covert timing channels

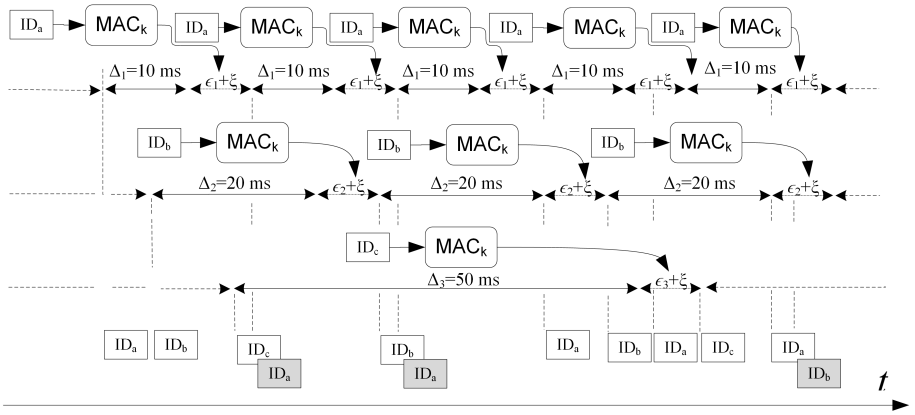

Fig. 3. Overview of the mechanism: frame arrival on the bus at delays $\Delta = 10, 20, 50ms$ with adjustment $\epsilon_i, i = 1..3$ and covert authentication delay ξ resulting from a cryptographic MAC

on CAN bus and this work. We try to extract comparative results in a unitary manner from TACAN [14] and INCANTA [13]. Both these works experiment with a channel created by altering the arrival time of a single ID which is used as a carrier. In particular, TACAN [14] uses a carrier ID broadcast at $10ms$ and encodes 1-bit of information in the arrival time while INCANTA [13] uses a carrier ID broadcast at $100ms$ and encodes 5-bits of information. To increase the data-rate, TACAN [14] also uses a least-significant bit channel by which the last bit of the data-field is modified, but this is not a timing covert channel and thus we do not include it in this analysis focusing only on the Inter-Arrival Time (IAT) version of TACAN. The bit-error rate (BER) is less than 1% for [14] with the mention that this value for the BER is obtained by approximating over 4 frames (otherwise the BER could top significantly higher at around 40%). For [13] the value of the BER is 1.75% and holds for a high-priority ID that easily wins arbitration. In terms of throughput, 22.5 bps are reported by [14] and we can compute a data-rate of 57 bps for [13] which despite the slower periodicity of the ID, i.e., $100ms$ in [13] vs. $10ms$ [14], uses a higher drift for the ID, i.e., $2^{20}ns$ with a tolerance of $20.000ns$ which leads to around 5 bits extracted from each message. Neither [14] nor [13] experiment with covert data carried by multiple IDs and it is unlikely that the performance extends linearly with the number of IDs due to increased unpredictability of the arrival time in case of lower-priority IDs. In contrast, in CANTO we use all the IDs from the bus for the timing covert channel. The optimal traffic allocation from CANTO allows us to achieve identical results for all IDs and thus a data-rate of around 5047 bps and a BER of 0.95% as we later show in the experimental section.

While there are not many related works on covert channels for CAN, there are several related works that are in close relation to our approach. Optimal traffic allocation with respect to the security payload has been targeted by a small amount of works dedicated to CAN security such as [20], [21], [22] and [23]. These works do not target the creation of a covert timing authentication channel, but they focus on optimization problems for CAN traffic.

Nonetheless, many recent research works have been focusing on using frame arrival time, i.e., the delays that we use to create a covert channel, in order to detect intrusions, e.g., [24] and [25]. By using Bloom filters [26], frame arrival time has been also combined with frame content to filter malicious activity in [27]. More recently, frame periodicity has been exploited to extract clock skews which is used to create a unique fingerprint for each device due to physical imperfections in oscillators in [28]. This sets room for physical fingerprinting of CAN nodes. The use of clock skews has been explored for fingerprinting computers for more than a decade by the work in [29] and not surprisingly it was also applied to smart-phones [30]. Unfortunately, identification mechanisms based on clock-skews are rendered ineffective by the fine grained control of time-triggered interrupts on embedded devices which allow them to potentially fake their clock-skews as demonstrated by [31]. All these works are exploiting the precision of the clock circuitry in the controller, which also stays at the core of our proposal here.

To save bits from the data-field, other works have suggested the use of the identifier field, i.e., [32], [33], [34] and [35], but this requires special care as the identifier field is critical for arbitration and also used for filtering purposes. An alternative to identify senders without compromising bits of the CAN frame is to use physical signal characteristics, e.g., [36], [37], [38], but these approaches may be vulnerable to small variations in bus impedance.

TABLE I
COMPARATIVE PERFORMANCE RESULTS FOR COVERT TIMING CHANNELS ON THE CAN BUS

Protocol	Throughput (single ID)	Throughput (all IDs)	BER	Security Level
TACAN-IAT [14]	22.5 bps	N/A	<1%	1 bit/frame
INCANTA [13]	57 bps	N/A	1.75%	<5 bits/frame
CANTO (this work)	36-366 bps	5047 bps	0.95%	3-5 bits/frame

II. BACKGROUND AND EXPERIMENTAL SETUP

This section gives a brief overview on delays and clock skews on the CAN network. Nonetheless we discuss limita-

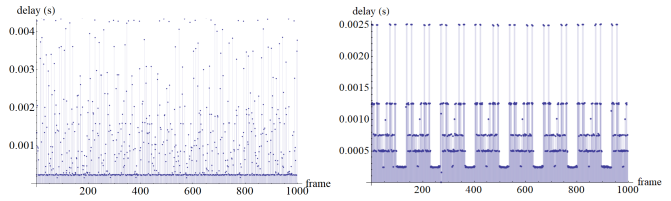

Fig. 4. Delays between frames: real-world car (left) vs. optimized traffic on our setup (right) at similar busload $30-40\%$

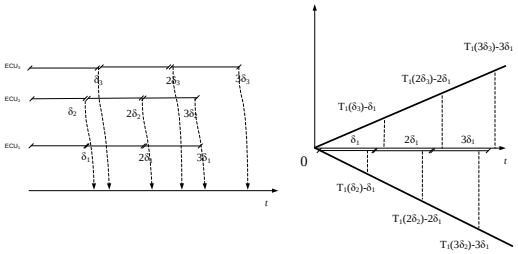

Fig. 5. Accumulation of clock skews for ECUs broadcasting at interval δ

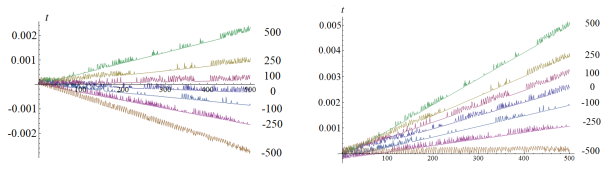

Fig. 6. Skews for a frame sent from an Infineon TC277 as recorded by an Infineon board (left) or from CANoe/VN CAN adapter (right) in [13]

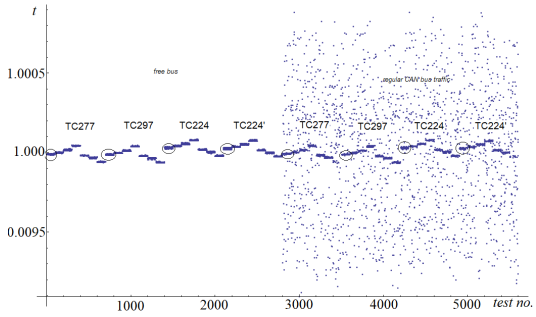

Fig. 7. Forced delays as recorded in [13] for a free bus (left) vs. a bus with regular network traffic (right)

tions in previous works on covert timing channels for the CAN bus. Then we describe the components of our setup.

A. Clock skews and limitations in previous work

In Figure 5 we show how clock skews accumulate when three ECUs are broadcasting at fix time intervals δ. While the delay δ is fixed, due to clock imprecision, the time measured at each ECU is in fact δ_1, δ_2 and δ_3 respectively. If the first ECU measures the arrival time for frames received from the second and third ECUs, the delays accumulate. The result is a slope which represents the skew of the clock. Figure 6 shows a graphical depiction for the delays measured on one Infineon board vs. CANoe in case of frames broadcast periodically by another Infineon board. The depiction is according to our previous work in [13]. Delays are forced at $\pm 100, \pm 250, \pm 500$ clock ticks (1 tick is $10ns$) and thus several slopes are visible in the picture.

The main limitation of our previous work on creating covert channels on the CAN bus was that existing traffic (poorly allocated) impedes the data-rate of the covert channel. Figure 7 shows the variation of delays recorded on four Infineon TriCore boards without (left) and with (right) existing network traffic according to [13] (delays are expressed as a fraction between the expected arrival time and recorded arrival time). In case of existing traffic, some of the frames arrive with significant delays making them indistinguishable for frames that are sent with random delays. These delays contribute to the false-positives of an intrusion detection mechanism. As we discuss and show in this work, traffic optimization is the only solution to this problem.

B. Worst-case arrival times

The worst-case arrival time of CAN messages is critical for assessing the viability of a particular allocation for message periods in a CAN network. Note that due to its ID-oriented arbitration, the arrival of low-priority IDs can be significantly delayed by IDs of higher priority. To serve us as a tool in proving the feasibility of our approach as well as for outlining the bandwidth constraints of the bus we now set a brief background on CAN schedulability.

For this, we will use the theoretical framework proposed in [39]. In particular we use the *busy period* t and *worst-case queueing delay* w of message m as defined in [39]:

$$t_m^{n+1} = B_m + \sum_{k \in hp(m) \cup m} \left\lceil \frac{t_m^n + J_k}{T_k} \right\rceil C_k \tag{1}$$

$$w_m^{n+1}(q) = B_m + qC_m + \sum_{k \in hp(m)} \left\lceil \frac{w_m^n + J_k + \tau_{bit}}{T_k} \right\rceil C_k \tag{2}$$

We keep the original notations from [39] and m is to be interpreted as the ID of the message which defines its priority. In the previous relations, B_m is the blocking delay caused by a lower priority message being transmitted, J_k is the queueing jitter of message k, T_k the period of message k, C_k the message transmission time, q the instance of message m and τ_{bit} the time to transmit a single bit on the bus. By $hp(m)$ we denote messages that have higher priority than m, i.e., messages with lower IDs. Both t and w are to be solved by recurrence over n (for more details we refer the reader to [39]) until the values of t and w converge, i.e., $t_m^{n+1} = t_m^n$ and $w_m^{n+1}(q) = w_m^n(q)$.

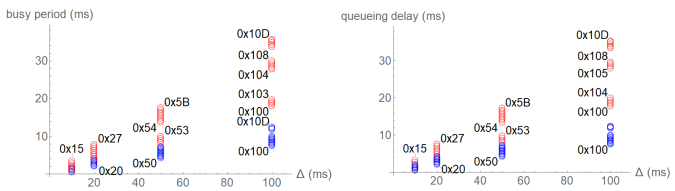

Fig. 8. Busy period and worst-case queueing delay as computed for the 40 IDs in our setup (blue) and impact of doubling the number of IDs (red)

We now show the results that we obtain by applying the previous methodology on our specific allocation with 40 IDs.

We selected the IDs and their periodicity to obtain a 40% bus-load that is characteristic to a real-world vehicle. The IDs have cycles of 10, 20, 50 and 100 ms (more evaluations of this allocation are given in the forthcoming section). We fix the following parameters that are characteristic for our setup: $\tau_{bit} = 2\mu s$ for the 500kbps CAN that we use, $C_k = 270\mu s$ which is the duration of the longest message on the bus (according to [39] this is computed as $(55 + 10B)\tau_{bit}$ where B is the number of bytes in the message) and $B_m = 270\mu s$ except for the ID with the lowest priority where $B_m = 0$. Figure 8 shows the busy period on the left and the worst-case queueing delay on the right for each of the 40 IDs (blue circles). The values for some of the IDs are depicted as labels but these are of little concern, the values are simply allocated such that faster IDs have higher priority and are consecutively numbered for the same periodicity so that that we could visually trace them easier. Then we depict with red circles the evolution of these when the number of IDs is doubled, e.g., the case in which we need to send additional frames with authentication data. In the original instantiation with 40 IDs, the busy period of the bus will be between 0.2–1.2ms for the IDs broadcast at $10ms$ and 5.4–8ms for the IDs at $100ms$. The queueing period stays in the range of 0.52 – $11.96ms$. A drift of $11.96ms$ may be somewhat high even for frames that have a period of $100ms$. However, when doubling the number of IDs, the busy period and worst-case queueing delay increase in the range of 0.52–34.06ms, that is by a factor of 3 in the worst case. By doubling the bus-load, the worst case arrival times increases higher than the expected twofold which suggests that adding additional frames is hardly an alternative. Optimizing traffic allocation will also help in this respect.

C. Setup components

We implement and evaluate optimizations on traffic allocation using an AURIX TC224 TFT Application Kit. The development board features a TC224 32-bit TriCore CPU that runs at frequencies up to 133 MHz and provides 1MB of FLASH memory and 96kB of RAM memory. The CAN frames transmitted by our TriCore-based implementation are recorded using CANoe, a software tool used for analyzing and testing of automotive networks. To achieve this, the CANoe running PC is interfaced with the development board through an VN CAN to PC adapter as depicted in Figure 9. The recorded traces were analyzed offline using Mathematica.

Since, according to the described mechanism, CAN frames have to be transmitted in specific time slots, nodes need to implement a time keeping functionality. We implemented this on the TC224 using the Capture/Compare Unit 6 Timer (CCU6) module which was configured to trigger an interrupt every 1 μs as a base tick for our local clock. In several experiments, we used the last 7 bits of a computed MAC value to represent the authentication delay. This delay is added to the message cycle time plus ϵ, and when enough CCU6 Timer's ticks have passed, the message is sent on the bus.

All of the message data bytes, configured message cycle times and the selected ϵ values for each message were configured in the MultiCAN+ module. The MultiCAN+ module is also responsible for transmitting the frame data to the CAN transceiver with the specified baudrate of 500 kbps.

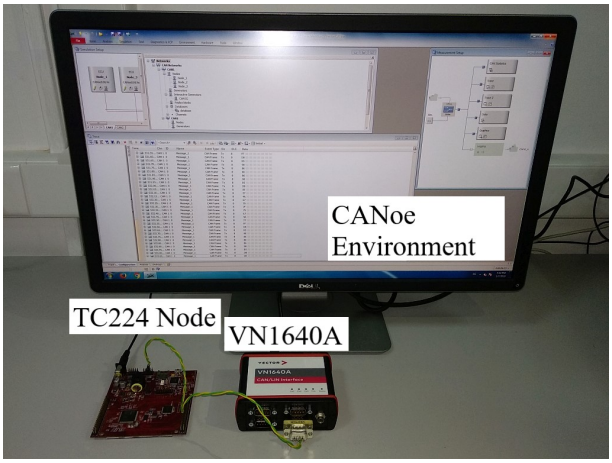

Fig. 9. Experimental setup used for generating and recording CAN traffic according to the proposed mechanisms

After performing the initial hardware setup, the initial MAC values are calculated for each message. During runtime, based on the counter value incremented in CCU6's timer, the cycle times for all frames and the ϵ values, each frame is sent, but with a small delay as already described. The frame delivery will take place after ξ ticks of the CCU6 Timer have expired. After each frame delivery, the message counter is incremented and a new MAC value is calculated based on the message data and the message counter.

III. Optimizing traffic allocation

This section addresses the optimization algorithms that we use. Traffic allocation is essential for achieving a satisfactory data-rate on the covert channel. We design and discuss four algorithms for traffic allocation and prove their effectiveness by both theoretical models/simulation and experimental data. We use two of these algorithms in the next section and implement the covert channel over optimized CAN-bus traffic.

A. Problem statement

We consider a set of n pairs $\{(\mathsf{id}_1, \Delta_1),\ (\mathsf{id}_2, \Delta_2),..., (\mathsf{id}_n, \Delta_n)\}$, each pair being formed by a CAN identifier and the cycle time (periodicity) corresponding to the identifier. If on-event frames exist, a distinct mechanism should be used, this situation however is out of scope for our work. Further, let $\{(\mathsf{id}_i, T_1^i), (\mathsf{id}_i, T_2^i), ..., (\mathsf{id}_i, T_l^i)\}$ be the set of identifier-timestamp pairs where timestamp $T_j^i, \forall j = 1..l$ is the time at which id_i was received on the bus. Ideally, $T_{j+1}^i - T_j^i = \Delta_i, \forall i = 1..n, j = 1..l$, which means that frames having the same identifier ID are received at periodicity Δ_i. In practice however, there are many reasons that impede a perfect arrival time. Besides clock drifts, i.e., the clock of sender and receiver nodes is not identical, delays may occur due to frames with overlapped sending time. Since CAN arbitration is non-destructive, there is no problem if two nodes try to send a frame at the same time. But the frame with the higher ID

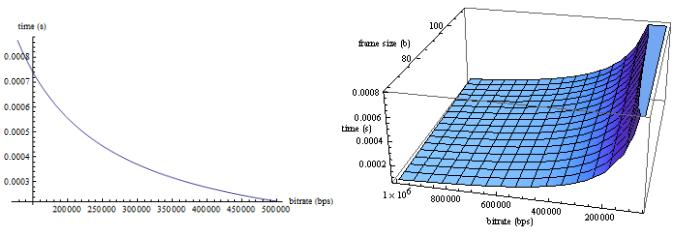

Fig. 10. Frame arrival time for a 64 bit data frame (left) and for a variable frame 0–64 bits (right) with data rates from 64kbps to 500kbps

loses arbitration and will be sent after the smaller ID which makes the arrival time drift from the expected Δ_i.

Frame arrival time. The time required for a frame to be transmitted on the bus depends on the size of the frame and data rate of the bus. Data rate can be up to 1Mbps in standard CAN, though lower data rates of 125-500kbps are commonly employed. The size of the frame varies due to the number of stuffing bits, i.e., one bit of reverse polarity is added after 5 consecutive identical bits (for a frame with 64 bits of data plus the header, a maximum of 24 stuffing bits can be added). The left side of Figure 10 shows the variation of frame arrival time in case of a 64 bits data frame which expands to 111 bits (without stuffing bits) and which may take as little as $100\mu s$ on 1Mbps or up to $900\mu s$ on a low-speed 125kbps bus (stuff bits not included). For a broader image, the right side of Figure 10 expands this calculation for variable size frames (0–64 bits) size and bus rates (64kbps–1Mbps).

Frame arrival time in real-world traces. In Figure 11 we depict the arrival time for frames scheduled at 10, 40 and $150ms$. The left side of the figure shows the delay between frames carrying the same ID and the right side the histogram distribution of the same delay. Even for the higher priority frame arriving at $10ms$, deviations of $400\mu s$ are common. For the $40ms$ frame deviations of $2-4ms$ are common and the situation is similar for the $150ms$ frame. In case of the $500ms$ frame, deviations of $10ms$ become common as well.

Such deviations from the expected arrival time exist and they clearly lower the bitrate of a covert timing channel. The deviations from the expected arrival time are directly influenced by local clocks and the priority of the message ID, but these can be circumvented by clever allocation of frame timings as we discuss next.

B. Optimizing frame scheduling

In the previously defined framework, if each frame is sent at multiples of $\Delta_i, i = 1..n$, the collisions on the bus between frame $i, j, \forall i, j = 1..n$ will occur at multiples of $lcm(\Delta_i, \Delta_j)$ (here lcm stands for the least common multiple of the two integers). This can be extended to any number of frames. In theory, all frames will collide on the bus at $lcm(\Delta_1, \Delta_2, ..., \Delta_n)$. Again, such collisions are non-destructive but they impede the covert timing channel since they cause additional delays that lead to deviations from the expected arrival time, prohibiting the extraction of covert bits from timing information. To avoid such collisions, we extend the frame scheduling set to $\{(\mathsf{id}_1, \Delta_1, \epsilon_1), (\mathsf{id}_2, \Delta_2, \epsilon_2), ..., (\mathsf{id}_n, \Delta_n, \epsilon_n)\}$ where $\epsilon_i, i = 1..n$ is a small drift added to the frame sending time. By our allocation, frames will be sent at intervals $k\Delta_i + \epsilon_i$ (rather than $k\Delta_i$). Our optimization problem consists in finding the values for $\epsilon_i, i = 1..n$ such that for a given set of delays $\Delta_i, i = 1..n$ no collisions will occur on the bus and moreover, the space between frames is maximized.

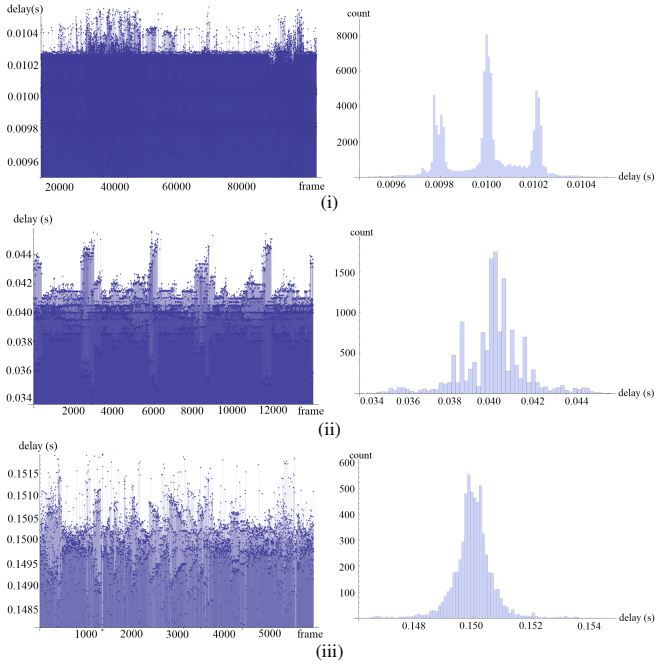

Fig. 11. Frame arrival time and histogram distribution of frame arrival time, for frames arriving at (i) 10ms, (ii) 40ms and (iii) 150ms delays

We use the following theoretical model to compute optimal frame allocation. Let the following n sets of traces corresponding to the n IDs broadcast over the CAN network:

$$\begin{cases} T_1 = \{(\mathsf{id}_1, \epsilon_1), (\mathsf{id}_1, \Delta_1 + \epsilon_1), ..., (\mathsf{id}_1, (l-1)\Delta_1 + \epsilon_1)\} \\ T_2 = \{(\mathsf{id}_2, \epsilon_2), (\mathsf{id}_2, \Delta_2 + \epsilon_2), ..., (\mathsf{id}_2, (l-1)\Delta_2 + \epsilon_2)\} \\ \qquad\qquad ..... \\ T_n = \{(\mathsf{id}_n, \epsilon_n), (\mathsf{id}_n, \Delta_n + \epsilon_n), ..., (\mathsf{id}_n, (l-1)\Delta_n + \epsilon_n)\} \end{cases} \tag{3}$$

Having the previous equations, we say that the *frame scheduling* is *complete* if $|T_1 \cup T_2 \cup ... \cup T_n| = |T_1| + |T_2| + ... + |T_n|$ where $|T_i|, i = 1..n$ denotes the cardinality of the set T_i. By this condition on the equality of the sets, we request that timings are distinct for all of the n frames.

Let $T^* = \{t_1, t_2, ..., t_n\}$ be the set containing all time stamps for all messages, we assume this set to be sorted in ascending order, i.e., the natural way in which frames are expected to arrive on the bus. To maximize the inter-frame space (IFS), we define a quality factor q under which we try to optimize the scheduling of the frames. We follow the natural intuition that the larger the IFS, the smaller the values $1/(t_i - t_{i-1})$ will be and thus their sum will be also smaller. We say that the *frame scheduling* is *optimal* if the following value is minimal:

$$q = \frac{1}{n} \sum_{i=2}^{n} \frac{1}{t_i - t_{i-1}} \tag{4}$$

Given that the IFS, i.e., $t_i - t_{i-1} \forall i = 2..n$, is measured in units of time, i.e., seconds, the q factor should be interpreted in units of frequency, i.e., s^{-1}. In what follows we discuss four variants of allocation algorithms that target the optimization of the quality factor q.

A practical instantiation. The subsequent optimization examples address the following frame periodicity vector which is based on existing CAN traffic from a real-world vehicle:

$$\overline{\Delta} = \{\underbrace{10, 10, ..., 10}_{\times 6}, \underbrace{20, 20, ..., 20}_{\times 8}, \underbrace{50, 50, ..., 50}_{\times 12}, \underbrace{100, 100, ..., 100}_{\times 14}\} \quad (5)$$

The vector $\overline{\Delta}$ corresponds to 6 IDs that have a cycle time of $10ms$, 8 IDs with a cycle of $20ms$, 12 IDs with $50ms$ and finally 14 IDs with a cycle time of $100ms$. The values in the vector represent the intended period for frames with the same ID, their arrival time will be only slightly affected by the delay on the covert channel which is of up to two hundred micro-seconds. In what follows we eliminate frame collisions on the bus by modifying the sending time with a small value $\epsilon_i, i = 1, n$ that does not affect the cycle time but only slightly shifts it as can be seen from relation (3).

Binary symmetric allocation. This is the simplest of the allocation algorithms that we use, it is very easy to implement and gives good results (we improve however on the inter-frame space with the next algorithms). In the binary symmetric allocation, we start with a bin size equal to the window size w which is the minimum of the delays $w = \min(\Delta_1, \Delta_2, ..., \Delta_n)$ and allocate all the values $\epsilon_i, i = 1..n$ to fit symmetrically in the interval $[0..w]$. For this we start with a bin of size w then we create new values ϵ_i by dividing each existing bin. That is, ϵ_1 is first placed at $w/2$, then for ϵ_2 and ϵ_3 two new bins are created at $w/4$ and $3w/4$, etc. Algorithm 1 shows the steps of the binary symmetric allocation. We assume the delays appear in $rdelays$ in ascending order (Δ_1 is missing from the list since it is already allocated to a default value $\epsilon_1 = 0$). Variable $blist$ is instantiated with the first delay Δ_1 (for which we allocate a default value $\epsilon_0 = 0$) and a second delay ∞ which is merely a placeholder for delimiting the bin which has size Δ_1. In step 4 we loop until an ϵ_i is generated for each value Δ_i. For this purpose, we create a new list $blist'$ in step 5 and loop in step 6 over all existing values in $blist$ to create a new value $(blist[i-1,1] - blist[i,1])/2$ that is added between $blist[i-1]$ and $blist[i]$ in the newly created list $blist'$. In step 15 $blist$ is replaced with $blist'$ at each iteration. At the end of the procedure $blist$ will contain all pairs $\{(\epsilon_1, \Delta_1), ..., (\epsilon_n, \Delta_n)\}$. A graphical depiction of the theoretical frame timings is in Figure 12. In Figure 13 we present the experimental results as measured from CANoe in when an Infineon node broadcasts frames with the corresponding timings. The theoretical results and experimental measurements are reasonably close. Differences exists as several frames are broadcast later resulting in a $2.5ms$ inter-frame delay. The reason for this is that the minimum inter-frame space is at $150\mu s$ which is also around the time needed to place a frame on the bus at 500 kbps. Due to computational delays on the controller, if the time-slot is missed, the frame will be sent at some later point missing the expected allocation on the bus. We conclude that $150\mu s$ for inter-frame space is somewhat too short.

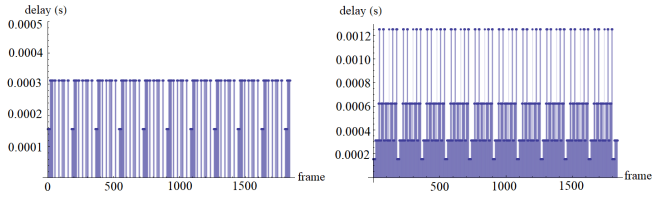

Fig. 12. Theoretical displacement of delays in case of binary symmetric allocation: detail for delays lower than $300\mu s$ and overall view up to $1.2ms$ for the first 2000 frames

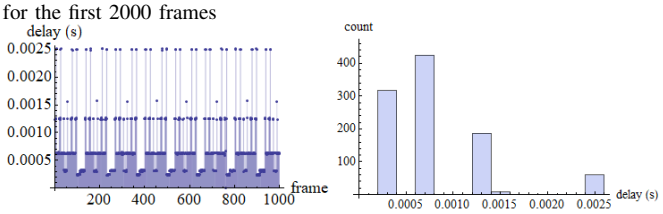

Fig. 13. Experimental measurements from CANoe of an Infineon node broadcasting after binary symmetric allocation: delays (left) and histogram distribution of delays (right)

Algorithm 1 Binary symmetric allocation

```
1: procedure BINARY ALLOCATION
2:     rdelays ← {Δ_2, ..., Δ_n}
3:     blist ← {(0, Δ_1), (Δ_1, ∞)}
4:     while size(rdelays) > 0 do
5:         blist' ← {blist[1]}
6:         for i = 2, i ≤ size(blist) do
7:             if size(rdelays) > 0 then
8:                 blist' ← append(blist', (blist[i − 1, 1]
                                  − blist[i, 1])/2, rdelays[1]))
9:                 blist' ← append(blist', blist[i])
10:                rdelays ← delete(rdelays, 1)
11:            else
12:                blist' ← append(blist', blist[i])
13:            end if
14:        end for
15:        blist ← blist'
16:    end while
17: end procedure
```

Randomized search allocation. The randomized search first creates a list of $\epsilon_i, i = 1..n$ that are equally spaced. A distance set to e where $e = \min(\Delta_1, \Delta_2, ..., \Delta_n)/n$ is a natural choice since at worst all frames will appear during the periodicity of the fastest frame from the bus. However, other values may be fixed for e. Then the $\epsilon_i, i = 1..n$ values are allocated randomly to each delay $\Delta_i, i = 1..n$. After ℓ iterations (each consisting in a randomized allocation) the best allocation is kept. The result improves with a higher number of iterations. Algorithm 2 shows the steps of the randomized search allocation. First, the value of e is computed in line 2, then values of the target epsilons are generated in line 3 and stored in $reps$. The current optimum q is set to ∞ in line 6 and the best allocation $reps''$ is set to void in line 7. Then we loop in line 8 for $max_{iterations}$. During each loop, the new values $reps'$ are set to a random permutation from $reps$ in line 9. Lines 10–16 compute the quality factor q for the new allocation.

Algorithm 2 Randomized allocation

```
1: procedure RANDOMIZED ALLOCATION
2:     e ← min(Δ_1, Δ_2, ..., Δ_n)/n
3:     reps ← {0, e, 2e, ..., (n − 1)e}
4:     delays ← {Δ_1, Δ_2, ..., Δ_n}
5:     i = 1
6:     q ← ∞
7:     reps'' ←⊥
8:     while i ≤ max_iterations do
9:         reps' = randomize(reps)
10:        T* ← {}
11:        for j = 1, i ≤ n, j = j + 1 do
12:            T_j ← {kΔ_j + reps'[j]; k = 1..⌊T/Δ_j⌋}
13:            T* ← append(T*, T_j)
14:        end for
15:        T* ← sort(T*)
16:        q' = (1/n) Σ_{i=2}^{|T*|} 1/(t_i − t_{i−1})
17:        if q' < q then
18:            reps'' = reps'
19:            q = q'
20:        end if
21:        i = i + 1
22:    end while
23: end procedure
```

For this, the set of timestamps $T_j, j = 1..n$ is generated for each delay, according to the corresponding ϵ_j of the current permutation. The timestamps are computed for a timeframe T (in our practical tests we set this to 1-10 seconds). Then the set of timestamps T^* is sorted and q is computed accordingly. If the result is better than for the previous optimum, i.e., line 17, the new result is stored in $reps''$, otherwise it is dropped. The randomized search gave somewhat better results than the binary symmetric allocation, but again the $250\mu s$ seems to be problematic for some frames (we improve on this with the next two algorithms).

Greedy allocation. In the Greedy allocation, for n delays, we first create bins that are equally spaced at distance $e = \min(\Delta_1, \Delta_2, ..., \Delta_n)/n$ (this is identical to the case of the previous algorithm). Then we allocate the delays in ascending order in such way that q is minimized. In Algorithm 3 we show the steps of the Greedy allocation. The algorithm starts by building the set of values $\epsilon_i, i = 1..n$ in a similar way to the randomized allocation described previously. Then it loops for each delay in line 7 and for each of the remaining values ϵ loops again in line 10 in order to find the optimum value for the current delay. Each of the selected values is tested against the optimization criteria in a similar manner to the randomized allocation. The index of the optimal value is stored in ind and this is removed from the ϵ values stored in $reps$ in line 21 such that only the remaining values could be allocated for the next delay Δ_i.

Greedy Multi-Layer. To circumvent the $250\mu s$ inter-frame space issues we modify the Greedy allocation to a *Multi-Layer Greedy* allocation in which frames at delay $\Delta_i, i = 1..n$ are allowed to be placed at any multiple of e that is smaller than Δ_i. This allows for a better expansion of the frames since frames at a larger Δ_i can benefit from a larger ϵ_i. We skip formalism for this algorithm to avoid overloading the paper. The theoretical and experimental results are in Figures 14 and

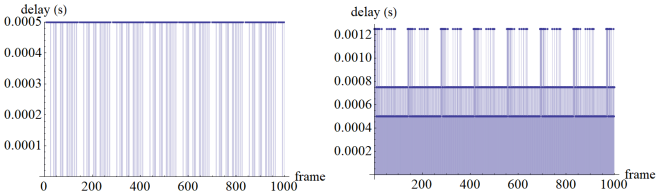

Fig. 14. Theoretical displacement of delays in case of Multi-Layer Greedy allocation: detail for delays lower than $300\mu s$ and overall view up to $1.2ms$ for the first 2000 frames

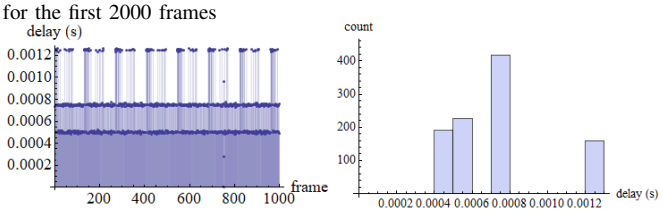

Fig. 15. Experimental measurements from CANoe of an Infineon node broadcasting after Multi-Layer Greedy allocation: delays (left) and histogram distribution of delays (right)

15. This time the results are almost identical and the inter-frame space is expanded to up to $500\mu s$.

GCD based allocation. We also try a Greatest Common Divisor (GCD) based allocation in which the frames are spaced by a delay, e.g., fixed at $500\mu s + \delta_i$ where $\delta_i, i = 1..n$, which is a multiple of $\gcd(\Delta_1, \Delta_2, ..., \Delta_n)$ subject to the condition that $500\mu s + \delta_i$ is smaller than Δ_i. The steps of the GCD based allocation are depicted in Algorithm 4. It starts from building a matrix M of $\max(delays)/G$ rows and $\min(delays)/\varepsilon$ columns, where G is the greatest common divisor of the delays and ε is the minimum allowable IFS. The matrix is initially filled by 1s. Then in step 7 the algorithm loops for all the delays searching for each of them an empty

Algorithm 3 Greedy allocation

```
1: procedure GREEDY ALLOCATION
2:     e ← min(Δ_1, Δ_2, ..., Δ_n)/n
3:     reps ← {0, e, 2e, ..., (n − 1)e}
4:     reps' ← {}
5:     delays ← {Δ_1, Δ_2, ..., Δ_n}
6:     T* ← {}
7:     for i = 1, i ≤ size(delays), i = i + 1 do
8:         opt = ∞
9:         ind = 1
10:        for j = 1, j ≤ size(reps), j = j + 1 do
11:            T*_aux ← T*
12:            T_j ← {kΔ_j + reps'[j]; k = 1..⌊T/Δ_j⌋}
13:            T*_aux ← append(T*_aux, T_j)
14:            T*_aux ← sort(T*_aux)
15:            opt' = (1/n) Σ_{i=2}^{T*_aux} 1/(t_i − t_{i−1})
16:            if opt' < opt then
17:                ind = j
18:                opt = opt'
19:            end if
20:        end for
21:        reps ← remove(reps, ind)
22:        reps' ← append(reps', ind)
23:        T_i ← {kΔ_i + reps'[i]; k = 1..⌊T/Δ_i⌋}
24:        T* ← append(T*, T_i)
25:    end for
26: end procedure
```

Algorithm 4 GCD-based allocation

1: **procedure** GCD-BASED ALLOCATION
2: $delays \leftarrow \{\Delta_1, \Delta_2, ..., \Delta_n\}$
3: $G \leftarrow \gcd(delays)$
4: $\varepsilon \leftarrow 0.5$
5: $reps \leftarrow \perp$
6: $M \leftarrow 1^{[1..\max(delays)/G][1..\min(delays)/\varepsilon]}$
7: **for** $i = 1, i \leq size(delays), i = i + 1$ **do**
8: $d \leftarrow delays[i]$
9: $j = 1$
10: $s = false$
11: **while** $s = false \& j \leq |M|$ **do**
12: $k = 1$
13: $aux = M[j]$
14: **while** $k \leq |aux| \& aux[k] = 0$ **do**
15: $k = k + 1$
16: **end while**
17: **if** $k \leq |aux|$ **then**
18: $l = d/G$
19: $a = k$
20: $s = true$
21: **while** $a \leq |aux|$ **do**
22: **if** $aux[a] \neq 0$ **then**
23: $aux[a] = 0$
24: $a = a + l$
25: **else**
26: $s = false$
27: **end if**
28: **end while**
29: **if** $s = true$ **then**
30: $M[j] = aux$
31: $reps = append(reps, (j-1)\varepsilon)$
32: **end if**
33: **end if**
34: $j = j + 1$
35: **end while**
36: **end for**
37: **end procedure**

row in the matrix M. The search is done in the loop from step 11 starting from row $j = 1$ (initialized in step 9) and continuously increases j in step 35 until a row without 0s is found. To test that the current row has non-zero values, the loop in step 14 goes until the end of the current line aux. If the resulting k is smaller than the length of the line $|aux|$, then starting from step 17, the line is filled with zeros at $l = d/G$ steps (where d is the current delay). The result for the current delay is placed in $reps$ in line 31. For a crisper view, the resulting drifts ϵ_i (expressed in milliseconds) for the delays in relation (5), that correspond to the 40 IDs in our experimental setup, are as following:

$$\epsilon_{i:1,40} = \{0, 0.5, 1.0, 1.5, 2.0, 2.5, 3.0, 13.0, 3.5, 13.5, 4.0, 14.0, 4.5, 14.5, 5.0, 15.0, 25.0, 35.0, 45.0, 5.5, 15.5, 25.5, 35.5, 45.5, 6.0, 16.0, 26.0, 36.0, 46.0, 76.0, 86.0, 96.0, 6.5, 16.5, 26.5, 36.5, 46.5, 56.5, 66.5, 76.5\} \quad (6)$$

The allocation provides results similar to the case of the Multi-Layer Greedy allocation. The theoretical and experimental measurements are in Figures 16 and 17 for a minimum inter-frame of $0.6ms$ and a $2.2ms$ maximum. When the minimum inter-frame space was lowered to $0.5ms$, similar to the case of the Greedy ML, the maximum increases to $4ms$.

In Table II we summarize a comparison between the four optimization algorithms. The main drawback with the first

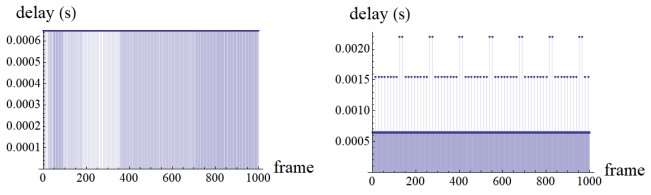

Fig. 16. Theoretical displacement of delays in case of GCD at $0.6ms$ allocation: detail for delays lower than $300\mu s$ and overal view up to $1.2ms$ for the first 2000 frames

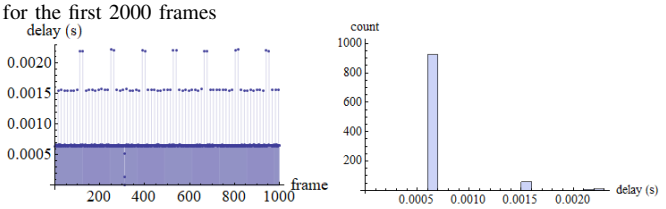

Fig. 17. Experimental measurements from CANoe of an Infineon node broadcasting after GCD allocation at $0.6ms$ inter-frame space: delays (left) and histogram distribution of delays (right)

TABLE II
COMPARISON OF ALLOCATION ALGORITHMS

	Completeness	q-Factor (s^{-1})	Min IFS (ms)	Max IFS (ms)
Binary Sym. Search	✓	2.37	0.15	2.5
Randomized Search	✓	2.51	0.25	3.75
Greedy Search	✓	2.39	0.25	2.5
Greedy ML Search	✓	1.50	0.5	1.25
Circular GCD	✓	1.86	0.5	4

three algorithms is that they leave a minimum inter-frame space of only $150\mu s$ - $250\mu s$ which is problematic since it may be smaller than the transmission window for a single frame (this may cause delays that pile-up and compromise the data-rate of the covert channel). The multi-layer Greedy and circular GCD returned the best results with a minimum IFS of $500\mu s$ which is sufficient even to accommodate the transmission of an additional frame. The value of the maximum IFS is not an optimization criterion in our algorithms and does not affect the covert channel (which exploits delays less than the minimum IFS). However, if the minimum IFS is closer to the maximum IFS, the busload will be more uniform and this may be preferable for other constraints, e.g., real-time response. Since implementation and computational costs are low for both the Greedy ML and circular GCD algorithms, we consider that any of them is suitable for a practical scenario. The results from Greedy ML offered a more uniform busload nonetheless.

IV. PROTOCOL AND RESULTS

In this section we give an outline of the protocol, then we discuss practical results on the covert timing channel that carries authentication tags.

A. Main protocol

In the current work we are focused on bus optimization for achieving a maximum capacity for the covert channel. The

protocol for sending and receiving frames is not distinct from our previous proposal [13] named INCANTA since we use the same kind of covert channel based on the drift of the frame from the expected arrival time. A minor difference is that to each delay $\Delta_i, i = 1..n$ we add the corresponding value $\epsilon_i, i = 1..n$ resulting from the optimization algorithm. For consistency, we keep the same name and description for our protocol INCANTA (INtrusion detection in Controller Area Networks with Time-covert cryptographic Authentication). INCANTA consists in the following set of actions that are to be followed by each node:

1) $\mathsf{SendCyclic}(\mathsf{id}_i, \mathrm{m})$ is the procedure triggered at some fixed delays $k\Delta_i + \epsilon_i$ for a frame with identifier field id_i and message content m at which the responsible sender ECU computes the authentication tag $tag = MAC_{\mathsf{sk}}(k, \mathsf{id}_i, \mathrm{m})$ by using a cryptographic MAC. Here k is a counter that is incremented for each new message that is sent with the same identifier. The sender then sets $\mathsf{T} = \lfloor tag \rfloor_\ell$ and performs a wait operation $wait(\mathsf{T})$ then broadcasts message $(\mathsf{id}, \mathrm{m})$,
2) $\mathsf{RecCyclic}(\mathsf{id}_i, \mathrm{m})$ at which the k^{th} instance of a message with identifier id_i is received. Let time t_k be the time at which the message is received, the receiver computes $tag = MAC_{\mathsf{sk}}(k, \mathsf{id}, \mathrm{m})$ and $\mathsf{T}_k = \lfloor tag \rfloor_\ell$ then checks if $|\mathsf{t}_k - \mathsf{t}_{k-1}| - (\Delta_i + \mathsf{T}_k - \mathsf{T}_{k-1})| \leq \rho$ and if this fails it drops the frame and reports an intrusion otherwise it considers the frame as genuine.

We consider that a shared secret key sk exists on each ECU from the CAN bus. We do not discuss how this key is shared since this is addressed by several other works. Moreover, we consider that the frame-scheduling set $\{(\mathsf{id}_1, \Delta_1, \epsilon_1), (\mathsf{id}_2, \Delta_2, \epsilon_2), ..., (\mathsf{id}_n, \Delta_n, \epsilon_n)\}$ is available to all genuine ECUs on the network. The delay is computed as the difference between two consecutive timestamps for the same ID in order to remove potential clock-skews. Indeed, by experimental measurements we did determine that clock skews may impede correct identification of delays. Concrete values for practical instances of the scheme are discussed next.

As already stated in the introduction, the covert channel that we propose should be viewed as a complementary measure for increasing security. To avoid replay attacks, mounted by an intruder based on the analysis of previously recorded delays, the use of freshness parameters must be re-enforced outside the covert channel. Current AUTOSAR specifications [10] require the use of a freshness parameter, of unspecified length in Profile 1 and 64 bits in Profile 3, that is further truncated to 8 bits in SecOC Profile 1 and 4 bits in SecOC Profile 3. In particular, in our implementation we used a counter for each ID that is further embedded in the frame and consequently used in the computation of the MAC which changes the forced delay on the covert channel. The size of the counter inside the frame can be kept in the recommended 4-8 bits range, and to avoid replays over a longer runtime, larger counters should be used. For example, 24 bits will be sufficient for 12 hours of runtime in case of an ID sent at $10ms$ and this counter will be truncated inside each frame. Thus we assume that only the 4-8 least significant bits are sent inside the frame for synchronization purposes, while each node maintains a larger, e.g., 24 bit counter for each ID.

A fast instantiation of the MAC is provided by the CMAC-AES which costs only 16-55μs on the Infineon controllers from our setup. The computation of the MAC should be done as soon as the data is ready for transmission and the time required to compute the MAC must not impede the delays established for the covert channel. The easiest way to achieve this is to include only the ID and the counter in the MAC since these are known in advance and thus the MAC can be computed at any time and stored in a buffer (note however that this does not hold as a security guarantee for the rest of the data-field).

B. Adversary model

We consider the regular type of Dolev-Yao adversary that has full control over the communication channel. Also, we consider that legitimate nodes from the bus which are in possession of the correct frame scheduling and MAC keys can be trusted and the adversary is an external node. This is realistic for automotive scenarios where component manufacturers can be trusted and most of the attacks reported so far were caused by outside interventions. We also consider attacks that remove a genuine node from the bus to be harder to achieve in an automotive-based scenarios since this requires either physical intervention or placing the node in the bus-off state. The later possibility has been recently demonstrated in [40] by exploiting the CAN error management system but it requires active error flags (which are visible on the bus) and the genuine node will remain in the bus-off state only for a fixed period of time. Moreover, removing a node will likely result in losing many functionalities from the car and many IDs from the bus, a behaviour which will likely be immediately recognized by the remaining ECUs. If the adversary cannot remove the genuine node from the bus it is very likely that the only effect of the adversary's intervention will be a DoS (Denial-of-Service) since the injected adversarial frames will likely cause delays on the genuine frames. Due to the design of the CAN bus, a DoS attack is always easy to mount on the bus. This has been already proved by works such as [40], [41], [42] which exploit either the physical properties of the bus, e.g., the fact that dominant bits overwrite recessive bits and thus legitimate frames can be impaired, or the error-handling mechanism of CAN which may place nodes in the bus-off state. Since our solution is dependent on an accurate frame arrival time, such an attack may be somewhat easier to mount but traffic abnormalities will be easy to detect from bad timing information on the covert channel. Assuming that the adversary can by some mean target a genuine node or a specific ID and remove it from the bus, the adversary further has to guess the exact delay at which the genuine frame needs to be sent. The success rate of an adversary can be estimated synthetically as:

$$\gamma_{adv} = \frac{\rho}{2^\ell} \tag{7}$$

This equation models the expected scenario where an adversary can at best insert a frame at some random point that

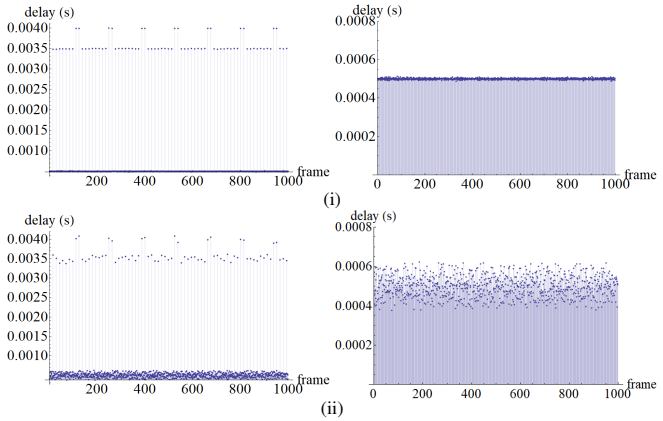

Fig. 18. Interframe delays (broadcast from Infineon TriCore node) in case of circular GCD optimization $\epsilon = 0.5$ without a covert channel (i) and with the covert channel in place (ii)

hopefully will match the expected delay. Here ρ is the delay tolerance for accepting a frame and ℓ is the security level (this are part of the protocol description that follows). The value of ρ has to be fixed based on experimental data, in the next section we show that a suitable value is $\rho = 5\mu s$ or even less. Given the worst case clock synchronization precision suggested in current standards [16] setting $\rho = 10\mu s$ may be also considered for a worst-case scenario.

C. Results with optimized traffic and a single sender

The experimental results that follow are based on optimized traffic with either the Greedy Multi-Layer or the GCD-based allocation. Since both algorithms yield a minimum IFS of $500\mu s$ the experimental results were similar with both allocations. Moreover, the $500\mu s$ IFS allows a free space of $230\mu s$ to implement the covert channel (considering a worst case time of $270\mu s$ to send a frame on the bus). This would allow for a truncated MAC of 7-8 bits to be encoded in the delay. In the experiments we determined that $\ell = 8$, which leads to a maximum delay of $255\mu s$, may still cause some frame overlaps. Indeed, the $255\mu s$ added to the maximum frame duration of $270\mu s$ may at time exceed the $500\mu s$ IFS. However, $\ell = 7$ results in a delay of at most $127\mu s$ and was perfectly safe at this IFS with no overlaps. Thus we can encode at most 7 covert bits in each frame, but not all will be recovered due to measurement imprecisions (we later determine the exact capacity of the covert channel). In Figure 18 we give an overview on the inter-frame delays on the bus with (ii) and without (i) the covert channel in place. Note that when the covert channel is in place, the delays vary randomly with the last 7 bits of a MAC, i.e., at most $127\mu s$. That is, the deviations around the expected value (i) have a randomized distribution (ii). Consequently, by observing past values of the delays the adversary cannot predict future ones, i.e., replay attacks are not possible. This becomes more evident in the details from the right side of the picture for the case of frames separated by only $500\mu s$

The experiments that we carried proved to be consistent for all the IDs, regardless of the delays at which they are sent, i.e.,

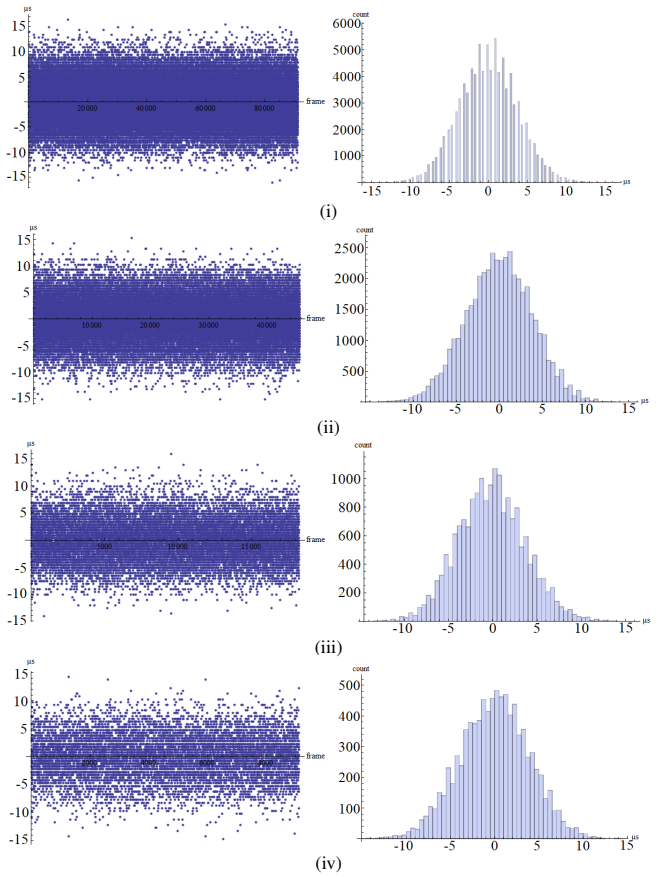

Fig. 19. Experimental measurements for covert authenticated frames from an Infineon TriCore node: deviation from the expected delays (left) and histogram distribution (right) for an ID sent at $10ms$ (i), $20ms$ (ii), $50ms$ (iii) and $100ms$ (iv)

$10, 20, 50$ or 100ms. The variation of the estimated delay was in the order of $\pm 10\mu s$ which is consistent with the time of 5 CAN bits (at 500kbps the duration of one bit on the CAN bus $\approx 2\mu s$). This variation may be due to the variation in frame length due to the number of stuffing bits which differs (we improve on this next).

Figure 19 shows the delays and their histogram distribution for frames broadcast at at $10ms$ (i), $20ms$ (ii), $50ms$ (iii) and $100ms$ (iv). Note that there are fewer samples as the delay increases. The deviation from the expected arrival time remains in the aforementioned range of $\pm 10\mu s$ for all frames and IDs. This is a good result considering the busload which is identical to real-world operation of the CAN bus. To go even further, we have also taken the frame length into account and achieve a better match as depicted in Figure 20. The minimum error for all messages was at $-4.62\mu s$ and the maximum at $4.87\mu s$ which means that at a $5\mu s$ tolerance all genuine messages will get the intended delay.

In Table III we give the average rate for the true negatives given the $2, 3$ or $4\mu s$ tolerance bound. We depict both the success rate for the genuine frames, i.e., γ_{ecu}, and the adversary advantage γ_{adv} in percents to facilitate interpretation of the data. The value of γ_{ecu} is extracted from the experimental data, i.e., a trace which covers around 1.2 million frames. The value is computed as the mean value of the acceptance

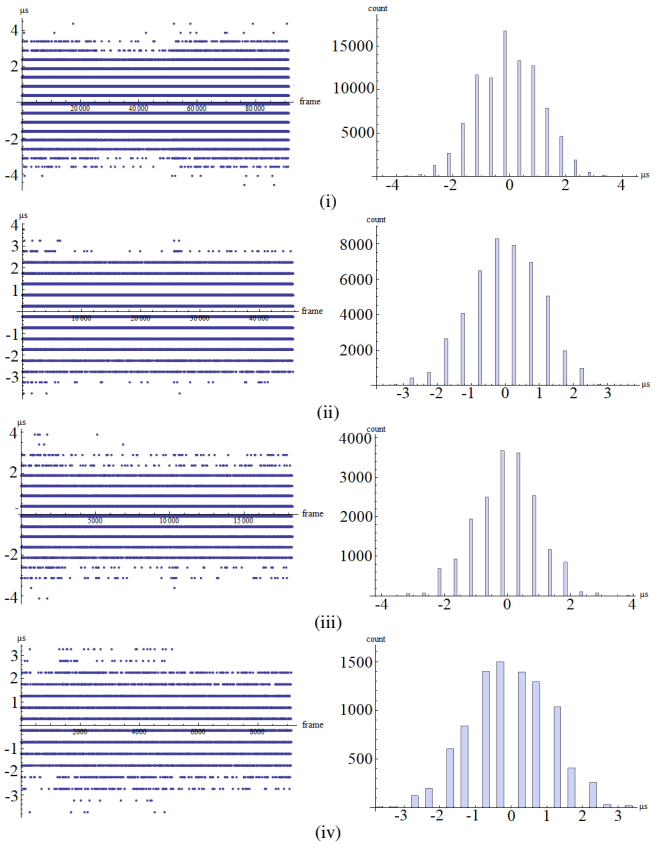

Fig. 20. Experimental measurements for covert authenticated frames from an Infineon TriCore node: deviation from the expected delays (left) and histogram distribution (right) for an ID sent at $10ms$ (i), $20ms$ (ii), $50ms$ (iii) and $100ms$ (iv)

rate taken over all messages corresponding to the 40 IDs. Concretely, depending on the ID, the $2\mu s$ error covered between $91.4\% - 95.2\%$ of the legitimate frames, while the $3\mu s$ and $4\mu s$ covered $99.21\% - 99.93\%$ and $99.96\% - 100\%$ respectively. The adversary advantage γ_{adv} is synthetically computed based on relation (6) at $\rho = 10$ and $\ell = 7$. We also extend these results for the case of multiple frames, i.e., over k consecutive frames. The acceptance rate of k legitimate frames is:

$$\gamma_{\blacklozenge}(k) = \gamma_{\blacklozenge}^{k}, \blacklozenge \in \{adv, ecu\} \tag{8}$$

Here k stands for the number of frames. Based on the data from Table III, in case when $\rho = 5\mu s$ all the genuine frames are accepted, while the chance of an adversary to inject a frame is less than 1 in a million.

D. The multi-sender case and noisy channels

For a more practical evaluation of the proposed mechanisms, we now extend our analysis for the case of multiple senders and noisy channels. We show that with proper parameter setup, both these scenarios can be addressed. The first scenario raises problems regarding synchronization and clock skew removal between senders. However, we obtain results similar to the single sender case after proper synchronization and skew removal. The second scenario is more problematic since

TABLE III
SUCCESS RATES (%) WITH TOLERANCE $\rho \in \{2, 3, 4, 5\}\mu s$ AT $\ell = 7$

ρ		$k=1$	$k=2$	$k=3$	$k=4$	$k=6$
$2\mu s$	γ_{ecu}	93.34	87.14	81.34	75.93	66.10
	γ_{adv}	3.1	0.09	0.003	0.00009	9.3×10^{-8}
$3\mu s$	γ_{ecu}	99.56	99.12	98.68	98.25	97.38
	γ_{adv}	4.7	0.22	0.01	0.0004	1.1×10^{-6}
$4\mu s$	γ_{ecu}	99.99	99.98	99.97	99.96	99.94
	γ_{adv}	6.2	0.39	0.02	0.001	5.9×10^{-6}
$5\mu s$	γ_{ecu}	100	100	100	100	100
	γ_{adv}	7.8	0.62	0.04	0.003	0.00002

optimal traffic allocation on the sender node meets existing in-vehicle traffic that is not-optimized, nor synchronized with the sender. While the performance is clearly lower, we show that even in this scenario we can attain a reasonable security level.

For the multi-sender scenario, we consider the case of two distinct senders and distribute half of the IDs to each of them. While more than two sender ECUs are generally present on an in-vehicle network, the synchronization results that we achieve for two senders should be extensible to any number of senders. In fact, the AUTOSAR specifications in [16], make it clear that the required $10\mu s$ synchronization error is for all time masters, gateways and even for slave nodes. For our testbed, we used one Infineon TC224 and one Infineon TC237 as the senders plus one VN1640A device to record traffic on the bus. The case of multiple senders on the bus can be efficiently addressed by proper synchronization and, as we later show, clock de-skewing techniques. As stated in the introductory section, clock synchronization is enforced by recent AUTOSAR standards not only for time-triggered buses but also for CAN [16]. In our testbed, it was sufficient to use a start-frame to announce the start of the broadcast for all nodes. We determined that, as expected, due to clock skews the two nodes lose synchronization after a few seconds of runtime. Figure 21 shows the effect of synchronization loss for a frame ID with a $10ms$ cycle time. As the clocks of the two sender nodes diverge, overlaps between frames start to appear making the frame shift with up to $\approx 200\mu s$ which is the duration of a frame on the bus (this is visible in part (i) of Figure 21). Note that this is in fact the blocking delay B_m from our previous worst-case analysis. A detailed view of the $[-10\mu s, 10\mu s]$ range, illustrated in part (ii) of the same figure, shows that most of the frames still arrive in time. We also depict the histogram distribution of the deviations in the $[-10\mu s, 10\mu s]$ range and in the $[-200\mu s, -10\mu s] \cup [10\mu s, 200\mu s]$ range in parts (iii) and (iv) of Figure 21. We suspected that these abnormalities are caused by the skew of the clocks and the definitive proof came after performing a skew adjustment. That is, we determined that the clock of the Infineon TC237 needs to be adjusted by a factor of 0.999965645 to match the clock of TC224. The slope of the clock, i.e., the skew, was determined programmatically, techniques for skew adjustment are well known in the literature, see [43]. To avoid precision loss due to float-to-integer conversions or causing too much computational overheads, we used the following correction in our code `ctime = ctime - 3.4355*(ctime/100000)` triggered at each $100ms$.

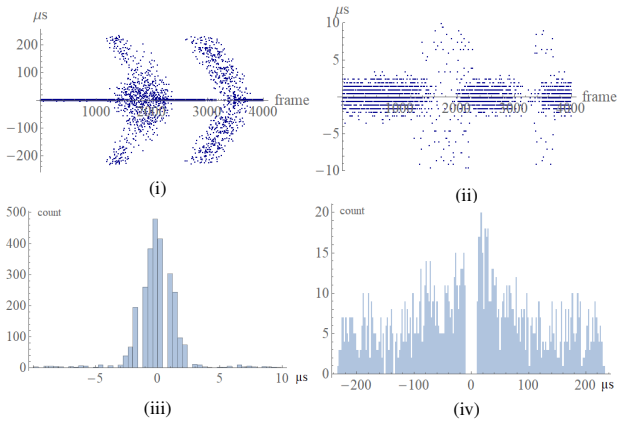

Fig. 21. Gaps due to synchronization loss in the expected arrival time of an ID (i), detailed view in the $[-10\mu s, 10\mu s]$ range (ii), histogram distribution of deviations in the $[-10\mu s, 10\mu s]$ range (iii) and histogram distribution of deviations in the $[-200\mu s, -10\mu s] \cup [10\mu s, 200\mu s]$ range (iv)

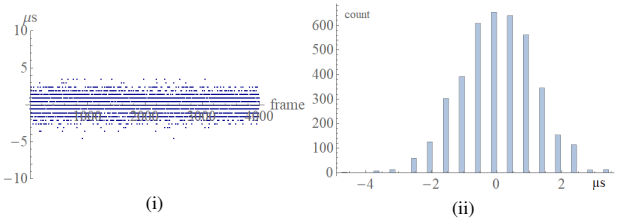

Fig. 22. The same ID after de-skewing: deviations from the arrival time (i) and their histogram distribution (ii)

Note that this line of code adjusts the clock by $3.4355\mu s$ at each $100ms$. The adjustment value comes from the skew coefficient 0.999965645 which would require an adjustment of $1 - 0.999965645 = 0.000034355$s each second, i.e, $34.355\mu s$ for each second or equivalently $3.4355\mu s$ at each $100ms$. The result after de-skewing is shown in Figure 22. By adjusting at $100ms$ intervals, the synchronization was almost perfect. Only a slight biasing of the distribution to the left side can be noticed in the histogram (note the -4 deviation to the left compared to the $+2$ to the right). Even with this deviation, almost all of the arrival times are kept in the expected range with a $\pm 5\mu s$ tolerance, similar to the case of a single sender. The synchronization error of $10\mu s$ is taken into account by the intrusion detection algorithm with the help of the tolerance parameter ρ. The traffic optimization algorithms in Section III, do not have to account for the $10\mu s$ error since they already space frames by 150-500μs, which is much larger than the synchronization error.

Table IV summarizes the results in distinct scenarios that we tested. The first row presents the single sender case and is identical to the results from Table III. Note that we do not include the values regarding the adversary success rate since these are identical to the ones in Table III as we use the same security parameter $\ell = 7$ and tolerances $\rho \in \{2, 3, 4, 5\}\mu s$. Then we proceed in rows two and three of the table to the case of two senders with optimized traffic allocation and no synchronization, then with proper synchronization and no skew correction. The results are somewhat similar: the absence of the skew correction has in the long run almost the same effects as the loss of synchronization, i.e., only 70–80% of the frames arrive in the $\pm 5\mu s$ tolerance. Finally, the fourth row of the table shows the results with optimized traffic allocation, proper synchronization and skew correction. This time the results are very close to the single sender case which proves that proper synchronization and skew correction solves the case of multiple senders.

The last two rows of Table IV show the results for a single sender over traffic recorded in a real-world vehicle that occupies 18% and then 36% of the bus. This scenario corresponds to the case when one retrofits an existing CAN bus with a sender capable of covert communication. Our sender node is responsible for an additional 18% busload, thus in the first case half of the traffic from the bus is optimized and the rest is not while in the later only one third of the traffic is optimized. In case of $\rho = 5\mu s$ the acceptance rate drops from 100% to 65% and then to 38%. This should not be surprising since the unoptimized traffic mixes at random with our allocation.

While the results with traffic overlaps do not look so well, the success rate of legitimate nodes is still much higher than that of the adversary. This suggests that by proper amendments of the scheme we can still separate between legitimate traffic and adversarial interventions. One straight-forward approach is to account for successful arrival of at least k-out-of-n frames (rather than account for every single frame) which follows the binomial distribution and leads to the following success rates for legitimate nodes and adversaries:

$$\gamma_{\blacklozenge}(k, n) = \sum_{l=k}^{n} \binom{n}{l} \gamma_{\blacklozenge}^{l} (1 - \gamma_{\blacklozenge})^{n-l}, \blacklozenge \in \{adv, ecu\} \quad (9)$$

By properly setting parameters k and n we can obtain very good separation rates. Table V shows the results by setting $n = 24$ and $k = 6..14$ in case of the results from the scenario with overlaps over 18% vehicle bus traffic. The $\pm 5\mu s$ tolerance brings more than 99.99% acceptance rate for legitimate frames and the chance for an intrusion not being detected is 0.8%. Thus, under normal circumstances at least $k = 6$ out of $n = 24$ frames would arrive on time while in case of an adversarial intervention k will be significantly lower and the attack detected. Frames should be continuously monitored and all of them accepted as long as k does not drop below 6 too often, e.g., more than 0.01% of the cases, otherwise an intrusion should be signaled and frames that do not arrive on time discarded (if the discarded frames will unavoidably include legitimate frames that cannot be separated, the adversary will at best cause a DoS). In case of the overlap with existing 36% vehicle bus traffic, we need to set $n = 48$ and $k = 8$ to get a 99.97% acceptance rate of legitimate frames and a 3.13% rate of undetected intruder activity. We believe that these are still good detection accuracies considering that our covert channel is overlapped with some arbitrary, existing in-vehicle traffic which adds significant noise to the channel.

E. Channel data-rate, security level and influence on worst-case arrival times

TABLE IV
SUCCESS RATES (%) FOR LEGITIMATE FRAMES IN DIFFERENT SCENARIOS ($\rho \in \{2,3,4,5\}\mu s$ AND $\ell = 7$)

Scenario	ρ			
	$2\mu s$	$3\mu s$	$4\mu s$	$5\mu s$
Single Sender	93.34	99.56	99.99	100
Dual Sender (opt.)	59.19	71.29	77.76	80.25
Dual Sender (opt./sync.)	69.38	76.40	77.56	77.89
Dual Sender (opt./sync./de-skew)	85.07	95.52	98.68	99.68
Single Sender (on 18% car traffic)	34.30	50.81	61.70	65.81
Single Sender (on 36% car traffic)	17.75	28.02	35.70	38.70

TABLE V
SUCCESS RATES FOR K-OUT-OF-N SCHEME WITH TOLERANCE $\rho \in \{2,3,4,5\}\mu s$, $\ell = 7$, $n = 24$ OVER 18% CAR TRAFFIC

ρ		$k=6$	$k=8$	$k=10$	$k=12$	$k=14$
$2\mu s$	γ_{ecu}	88.2321	61.5313	28.7884	8.23784	1.35971
	γ_{adv}	0.00770	0.00004	1.2×10^{-7}	1.6×10^{-10}	1.2×10^{-13}
$3\mu s$	γ_{ecu}	99.7402	97.3359	86.4570	61.1860	29.7967
	γ_{adv}	0.06864	0.00087	5.4×10^{-6}	1.8×10^{-8}	3.1×10^{-11}
$4\mu s$	γ_{ecu}	99.9946	99.8746	98.5952	91.6052	71.2112
	γ_{adv}	0.30101	0.00689	0.00007	4.7×10^{-7}	1.5×10^{-9}
$5\mu s$	γ_{ecu}	99.9992	99.9727	99.5740	96.5006	83.8611
	γ_{adv}	0.89451	0.03255	0.00059	5.7×10^{-6}	2.9×10^{-8}

We now consider to evaluate the maximum capacity of the covert channel. The data-rate of noisy channels can be computed with the Arimoto-Blahut algorithm [44], [45]. For this purpose, we extracted the channel matrix which gathers the probability that a sender delay $\Delta' \in [0, 255]\mu s$ decodes into a measured delay $\Delta'' \in [0, 255]\mu s$ (this corresponds to the case of $\ell = 8$). Once the channel matrix was extracted from the measurements, we used a freely available Matlab implementation of the algorithm[1] and determined that channel capacity is ≈ 4.9 bits. Again, this suggests that the 24-bit security level can be achieved in six CAN frames. This data-rate can be also estimated from the results in Table III since by setting a tolerance to $10\mu s$ we get a noiseless channel. But now the 256 symbols are reduced to 25 and thus a data-rate of ≈ 4.6 bits (this is a bit lower than the channel capacity computed with Arimoto-Blahut algorithms which gives an upper-bound). To avoid overlaps during the IFS, we used a 7-bit micro-second delay and given the $10\mu s$ resolution at the receiver, we have a rate of $\log_2(127/10) = 3.66$ bits for each frame.

Since the frame rate in our experiment was about 1379 frames-per-second we can estimate the maximum data-rate at $1379 \times 4.9 = 6757$bps. Based on our more simple 7-bit encoding and extraction, this turns into $1379 \times 3.66 = 5047$bps (this is the effective data-rate of our protocol). Depending on the cycle time of the ID, i.e., from $100ms$ down to $10ms$, from each ID we can extract 36–366bps with an expected maximum of 49–490bps (based on the computed maximum data-rate of the channel). The data-rate can be further increased when more frames are on the bus. For example, in theory, by using the bus at 54% capacity with frames equidistantly spaced at $500\mu s$, given the $270\mu s$ duration of a frame on the bus, there are still $230\mu s$ for a covert timing channel which at $10\mu s$ synchronization error could give a maximum data rate of 9800 kbps (considering the noise on the channel that results from our measurements). As for the noisy channels that result from overlaps with existing in-vehicle traffic, the results from Table IV point to a reduction in the acceptance rate from 100% in case of the single sender to 65.81% and 38.70% when overlapping with bus traffic at 18% and 36% busload respectively. This would lead to a corresponding data-rate of 3321bps and 1953bps for the two noisy channels in our experiments. To sum up, data-rates of several kilo-bits seem to be achievable for a covert timing channel on the CAN bus.

[1]https://www.mathworks.com/matlabcentral/fileexchange/32757-channel-capacity-using-arimoto-blahut-algorithm

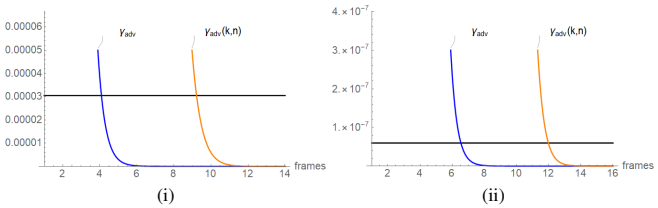

Fig. 23. Adversary success rate for multiple frames, i.e., $\gamma_{adv}(k)$, and k-out-of-n frames, i.e., $\gamma_{adv}(k,n)$, for: (i) $k \in [1,14]$ vs. a 2^{-15} security level and (ii) $k \in [1,16]$ vs. a 2^{-24} security level ($\rho = 5\mu s$)

Security level. In Figure 23 we depict an estimation for the adversary success rate compared to a 15-bit security level (left) and 24-bit security level (right). The 15-bit security level was chosen for comparison since it is the size of the CRC for CAN frames (the CRC is not resilient in front of adversaries but it worths as comparison), while the 24-bit security level is demanded by AUTOSAR [10]. We depict both the adversary success rate for multiple consecutive frames, i.e., $\gamma_{adv}(k)$, and for the k-out-of-n frames scenario, i.e., $\gamma_{adv}(k,n)$ for $n = 24$. The required security level of AUTOSAR can be reached in 6–7 frames in case of the first scenario. In 3 consecutive frames the security level is around 12 bits while for 6 consecutive frames it approaches the desired 24 bits for in-vehicle security. The authentication delay tops at $1.3ms$ for 3 frames and $6ms$ for 6 consecutive frames as depicted in Figure 24. This is just a worst case scenario since often the space between frames is $500\mu s$ and thus around $1.5ms$ or $3ms$ are to be expected. This is a very small delay for the 24 bit authentication level considering that no cryptographic operation is needed except for the regular MAC and the bus-load is not increased. For the second scenario, i.e., the k-out-of-n frame separation needed in case of the noisy channel (over 18% car traffic), we need around 12 correctly received frames out of 24 frames - which places the adversary advantage at 5.7×10^{-6} and that of legitimates frames at 96.50% providing good separation between the two (according to Table V). This roughly requires a period of 24ms to detect the intrusion which should be acceptable.

Impact on worst-case arrival time. We now consider to depict the influence of adding authentication delays on the busy period and worst-case arrival time of the frame. We underline that these effects will be visible only in case when the optimal traffic allocation cannot be preserved, otherwise the frames would arrive as scheduled. This may happen in case

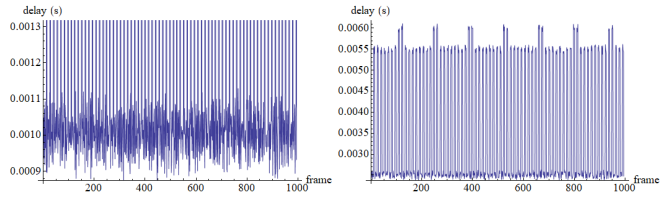

Fig. 24. Delay between 3 (left) or 6 (right) consecutive frames

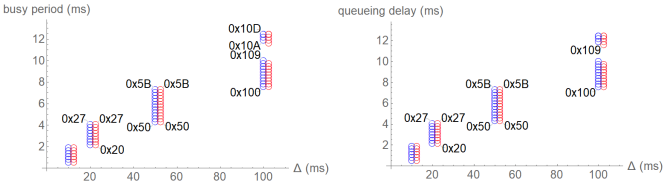

Fig. 25. Busy period and worst-case queueing delay as computed for the 40 IDs in our setup (blue) and impact of adding the authentication delay (red)

when synchronization is lost or if an adversary interferes with regular bus behaviour, i.e., by causing a DoS on the bus. The authentication delay that we add is essentially contributing to the queueing jitter of the message, i.e., J_k. In the experimental results, we use a delay of at most $127\mu s$ that is added to the legitimate sending time of the frame. This further results in a queueing jitter that is $127\mu s$ larger. Figure 25 shows the impact on the busy period (left) and the impact on the worst-case arrival time (right). Both remain largely unaffected which is to be expected since the added jitter is very small, i.e., $127\mu s$.

V. Conclusions

Our proof-of-concept implementation demonstrates that a covert channel which carries 3-5 bits per CAN frame can be implemented in a multi-sender scenario and high-end electronic control units are capable of maintaining the strict synchronization demands. Consequently, the 24-bit security level demanded by recent standards may be reached in about 6 frames in a covert manner. This does not imply that the solution would be easy to port in practice, our attempt here is to get closer to a practical upper bound and we hope that real world deployments could achieve more than the limited data-rate from previous research in [13] and [14] and comparable or even better results than the 5kbps reported here in CANTO. Our experimental analysis suggests that deviations from the expected arrival time of frames are mostly caused by unoptimized traffic rather than by the accuracy of the controller's clock. The four algorithms that we introduce for optimizing traffic allocation show clear advantages in this respect. The Greedy and Circular GCD optimizations give better results, with a minimum inter-frame distance of 500 μs which can further accommodate covert authentication. Over the optimized traffic, the expected arrival time drifts only in the $\pm 5\mu s$ range which roughly corresponds to the the time of 5 CAN bits at 500kbps. Further investigations may be needed to test the feasibility of the proposed procedures inside a real-world vehicle, but we do provide results from overlapping our covert channel on existing in-vehicle traffic as a start. Since modern time-triggered protocols and industry standards already demand synchronization in the order of $10\mu s$ in the worst case, we believe that covert channels may be implemented at the rigorous timing demands from our work. At the very least, we report a new experimental upper bound for such covert channels under optimized traffic flows, i.e., 3-5 bits per frame totaling 5kbps over the entire bus traffic.

Acknowledgement. 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).

References

[1] T. Hoppe and J. Dittman, "Sniffing/replay attacks on can buses: A simulated attack on the electric window lift classified using an adapted cert taxonomy," in *Proceedings of the 2nd workshop on embedded systems security (WESS)*, 2007, pp. 1–6.

[2] K. Koscher, A. Czeskis, F. Roesner, S. Patel, T. Kohno, S. Checkoway, D. McCoy, B. Kantor, D. Anderson, H. Shacham *et al.*, "Experimental security analysis of a modern automobile," in *Security and Privacy (SP), 2010 IEEE Symposium on*. IEEE, 2010, pp. 447–462.

[3] 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.

[4] C. Miller and C. Valasek, "Adventures in automotive networks and control units," *DEF CON*, vol. 21, pp. 260–264, 2013.

[5] O. Hartkopp, C. Reuber, and R. Schilling, "MaCAN-message authenticated CAN," in *10th Int. Conf. on Embedded Security in Cars (ESCAR)*, 2012.

[6] Q. Wang and S. Sawhney, "Vecure: A practical security framework to protect the can bus of vehicles," in *Internet of Things (IOT), 2014 International Conference on the*. IEEE, 2014, pp. 13–18.

[7] 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.

[8] 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.*, vol. 17, no. 8, pp. 2248–2261, Aug 2016.

[9] 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*. ACM, 2019, pp. 3–8.

[10] *Specification of Secure Onboard Communication*, 4th ed., AUTOSAR, 2017.

[11] T. Ziermann, S. Wildermann, and J. Teich, "CAN+: A new backward-compatible Controller Area Network (CAN) protocol with up to 16x higher data rates," in *Design, Automation and Test in Europe, DATE 2009, Nice, France, April 20-24, 2009*, 2009, pp. 1088–1093.

[12] A. Van Herrewege, D. Singelee, and I. Verbauwhede, "CANAuth-a simple, backward compatible broadcast authentication protocol for can bus," in *ECRYPT Workshop on Lightweight Cryptography*, 2011.

[13] 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.

[14] X. Ying, G. Bernieri, M. Conti, and R. Poovendran, "Tacan: Transmitter authentication through covert channels in controller area networks," in *Proceedings of the 10th ACM/IEEE International Conference on Cyber-Physical Systems*. ACM, 2019, pp. 23–34.

[15] *Requirements on Time Synchronization*, R19-11 ed., AUTOSAR, 2019.

[16] *Specification of Time Synchronization over CAN*, 4th ed., AUTOSAR, 2017.

[17] V. Berk, A. Giani, G. Cybenko, and N. Hanover, "Detection of covert channel encoding in network packet delays," *Rapport technique TR536, de lUniversité de Dartmouth*, vol. 19, 2005.

[18] Y. Liu, D. Ghosal, F. Armknecht, A.-R. Sadeghi, S. Schulz, and S. Katzenbeisser, "Hide and seek in time—robust covert timing channels," in *European Symposium on Research in Computer Security*. Springer, 2009, pp. 120–135.

[19] S. Cabuk, C. E. Brodley, and C. Shields, "Ip covert timing channels: design and detection," in *Proceedings of the 11th ACM conference on Computer and communications security*. ACM, 2004, pp. 178–187.

[20] C.-W. Lin, Q. Zhu, and A. Sangiovanni-Vincentelli, "Security-aware mapping for tdma-based real-time distributed systems," in *Computer-Aided Design (ICCAD), 2014 IEEE/ACM International Conference on*. IEEE, 2014, pp. 24–31.

[21] ——, "Security-aware modeling and efficient mapping for CAN-based real-time distributed automotive systems," *IEEE Embedded Systems Letters*, vol. 7, no. 1, pp. 11–14, 2015.

[22] C.-W. Lin, B. Zheng, Q. Zhu, and A. Sangiovanni-Vincentelli, "Security-aware design methodology and optimization for automotive systems," *ACM Trans. Des. Autom. Electron. Syst.*, vol. 21, no. 1, 2015.

[23] 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*, vol. 15, no. 2, pp. 1094–1104, 2018.

[24] M. R. Moore, R. A. Bridges, F. L. Combs, M. S. Starr, and S. J. Prowell, "Modeling inter-signal arrival times for accurate detection of can bus signal injection attacks: a data-driven approach to in-vehicle intrusion detection," in *Proceedings of the 12th Annual Conference on Cyber and Information Security Research*. ACM, 2017, p. 11.

[25] H. M. Song, H. R. Kim, and H. K. Kim, "Intrusion detection system based on the analysis of time intervals of can messages for in-vehicle network," in *Information Networking (ICOIN), 2016 International Conference on*. IEEE, 2016, pp. 63–68.

[26] B. H. Bloom, "Space/time trade-offs in hash coding with allowable errors," *Commun. ACM*, vol. 13, no. 7, pp. 422–426, Jul. 1970. [Online]. Available: http://doi.acm.org/10.1145/362686.362692

[27] B. Groza and P. Murvay, "Efficient intrusion detection with Bloom filtering in controller area networks," *IEEE Transactions on Information Forensics and Security*, vol. 14, no. 4, pp. 1037–1051, April 2019.

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

[29] T. Kohno, A. Broido, and K. C. Claffy, "Remote physical device fingerprinting," *IEEE Transactions on Dependable and Secure Computing*, vol. 2, no. 2, pp. 93–108, 2005.

[30] M. Cristea and B. Groza, "Fingerprinting smartphones remotely via icmp timestamps," *IEEE Communications Letters*, vol. 17, no. 6, pp. 1081–1083, 2013.

[31] 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*. IEEE Press, 2018, pp. 32–42.

[32] K. Han, A. Weimerskirch, and K. G. Shin, "A practical solution to achieve real-time performance in the automotive network by randomizing frame identifier," in *Proc. Eur. Embedded Secur. Cars (ESCAR)*, 2015, pp. 13–29.

[33] A. Humayed and B. Luo, "Using id-hopping to defend against targeted dos on can," in *Proc. of the 1st International Workshop on Safe Control of Connected and Autonomous Vehicles*. ACM, 2017, pp. 19–26.

[34] W. Wu, R. Kurachi, G. Zeng, Y. Matsubara, H. Takada, R. Li, and K. Li, "Idh-can: A hardware-based id hopping can mechanism with enhanced security for automotive real-time applications," *IEEE Access*, vol. 6, pp. 54 607–54 623, 2018.

[35] S. Woo, D. Moon, T.-Y. Youn, Y. Lee, and Y. Kim, "Can id shuffling technique (cist): Moving target defense strategy for protecting in-vehicle can," *IEEE Access*, vol. 7, pp. 15 521–15 536, 2019.

[36] 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*. ACM, 2017, pp. 1109–1123.

[37] 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.

[38] 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*. ACM, 2018, pp. 787–800.

[39] R. I. Davis, A. Burns, R. J. Bril, and J. J. Lukkien, "Controller area network (can) schedulability analysis: Refuted, revisited and revised," *Real-Time Systems*, vol. 35, no. 3, pp. 239–272, 2007.

[40] K.-T. Cho and K. G. Shin, "Error handling of in-vehicle networks makes them vulnerable," in *Proceedings of the 2016 ACM SIGSAC Conf. on Computer and Communications Security*. ACM, 2016, pp. 1044–1055.

[41] P.-S. Murvay and B. Groza, "Dos attacks on controller area networks by fault injections from the software layer," in *Proc. of the 12th Intl. Conf. on Availability, Reliability and Security*, 2017, pp. 1–10.

[42] A. Palanca, E. Evenchick, F. Maggi, and S. Zanero, "A stealth, selective, link-layer denial-of-service attack against automotive networks," in *International Conference on Detection of Intrusions and Malware, and Vulnerability Assessment*. Springer, 2017, pp. 185–206.

[43] S. B. Moon, P. Skelly, and D. Towsley, "Estimation and removal of clock skew from network delay measurements," in *INFOCOM'99, Proceedings of 18-th Annual Joint Conference of the IEEE Computer and Communications Soc.*, vol. 1. IEEE, 1999, pp. 227–234.

[44] S. Arimoto, "An algorithm for computing the capacity of arbitrary discrete memoryless channels," *IEEE Transactions on Information Theory*, vol. 18, no. 1, pp. 14–20, 1972.

[45] R. Blahut, "Computation of channel capacity and rate-distortion functions," *IEEE transactions on Information Theory*, vol. 18, no. 4, pp. 460–473, 1972.

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 research programs dedicated to automotive security funded by the Romanian Authority for Scientific Research and Innovation.

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.