CANASTA: Controller Area Network Authentication Schedulability Timing Analysis

Omolade Ikumapayi*, Habeeb Olufowobi†, Jeremy Daily‡, Tingting Hu§, Ivan Cibrario Bertolotti¶, Gedare Bloom*
*University of Colorado Colorado Springs, Colorado Springs, CO, USA
†University of Texas at Arlington, Arlington, TX, USA
‡Colorado State University, Fort Collins, CO, USA
§University of Luxembourg, Faculty of Science, Technology and Medicine (FSTM), Luxembourg
¶National Research Council of Italy, Institute of Electronics, Computer and Telecommunication Engineering (CNR-IEIIT), Turin, Italy
{oikumapa, gbloom}@uccs.edu, habeeb.olufowobi@uta.edu, jeremy.daily@colostate.edu, tingting.hu@uni.lu, ivan.cibrario@ieiit.cnr.it

***Abstract*—The Controller Area Network (CAN) dominates in-vehicle networking systems in modern vehicles. CAN was designed with low-latency and reliability as key features. Authenticity of a CAN frame was not considered in the design, thus, most in-vehicle network nodes inherently trust received messages as coming from a legitimate source. As a result, it is trivial to program (or hack) a network node to spoof traffic. Authentication is challenging for CAN and related protocols, such as SAE J1939, due to limited frame sizes and high bus utilization. Adding a message authentication code (MAC) as a separate message can unduly stress the real-time delivery of safety-critical messages. Although this stressor is well-known, the impact of authentication protocols on real-time message delivery in CAN has not yet been thoroughly examined. In this paper, we provide the first comprehensive analysis of real-time schedulability analysis applied to authentication schemes for CAN, CAN Flexible Data-rate (CAN FD), and CAN extra long payload (CAN XL). We formulate the response time analysis for addition of MACs and periodic transmission of MACs, and we examine their impact on two case studies and through evaluation with randomized schedulability experiments over a wide range of message sets.**

***Index Terms*—Controller Area Network, CAN FD, CAN XL, Response Time Analysis**

I. Introduction

THE ability for messages to meet real-time deadlines is a hallmark of the controller area network (CAN) protocol. Discovery of response time analysis (RTA) for CAN was a breakthrough that enabled vehicle designers to utilize bus bandwidth more effectively while ensuring that the delay incurred by safety-critical messages is bounded [1]–[3]. The expectation from analysing the worst-case response time of a CAN message is the reliability that the message will arrive at its destination in no more than the derived maximum time interval from the release time to completion of message transmission. While reliable delivery of messages is guaranteed with the CAN message acknowledgement process, provisions for authenticating the legitimacy of the message are missing from the protocol. Therefore, an application layer approach uses a message authentication code (MAC) to provide a cryptographic checksum on message transmission to ensure content integrity (no bits are modified) and origin integrity (no sender is impersonated) when the MAC is validated by the message recipient.

The AUTOSAR Specification of Secure OnBoard Communication (SecOC) [4] defines requirements for cryptographic authentication mechanisms that can be used to authenticate protocol data units (PDUs) for different wired bus architectures such as CAN or automotive Ethernet. In SecOC terminology, the authentication information comprises a Freshness Value (FV) and an Authenticator. The Authenticator is normally a cryptographic message authentication code (CMAC) using symmetric key encryption, although the specification allows for digital signatures under asymmetric keys too. AUTOSAR allows for the CAN designer to select the authentication schemes, key lengths, CMAC and FV lengths. The Authenticator is appended to a PDU, while the FV is synchronized among the communicating electronic control units (ECUs). A truncated FV may be appended to the Authenticator to assist in this synchronization, and it is also possible to omit an FV completely. The use of a CMAC and FV is intended to prevent attacks such as meddler-in-the-middle (MitM) and masquerade attacks [5]. The CMAC and FV provide cryptographic data integrity more than just a checksum used to detect an error. Additional bits reserved for the CMAC and FV for authentication enhance the strength of the encryption by reducing the probability of successful hash collisions and guessing attacks. Although key lengths of at least 128 bits are suggested, the specification identifies several *profiles* for truncation of MAC and FVs with the caveat that only MACs of 64 bits or longer may be considered sufficient to prevent guessing attacks. Table I summarizes the three profiles defined in SecOC (release 4.3.1). Note that profiles 1 and 3 both require 32 bits, while profile 2 fits in 24 bits but provides no protection against replay attacks due to the lack of an FV. So the truncated CMAC and FV are mainly useful in CAN only in case the data payload of the authenticated messages are less than or equal to 32 bits as well, i.e., DLC $\leq$ 4. Otherwise the

TABLE I: Truncated Bit Lengths of AES-128 MAC (most significant bits kept) and Freshness Values (least significant bits kept) in SecOC Profiles.

Name (Mnemonic)	CMAC Length	FV Length
Profile 1 (24Bit-CMAC-8Bit-FV)	24	8
Profile 2 (24Bit-CMAC-No-FV)	24	0
Profile 3 (JasPar)	28	4

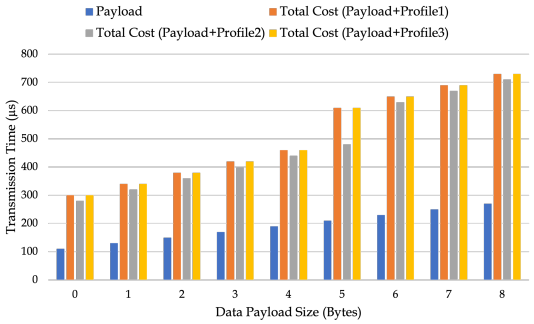


Fig. 1: Transmission times over 1 Mbps CAN bus with varying data payload sizes using SecOC profiles.

authenticator will require another message frame to be sent anyway, and so the full 64-bit data payload of the additional frame may as well be utilized. Without truncation, the CMACs typically will be 64- or 128-bits, while FVs are at most 64 bits.

Fig. 1 shows how the transmission times of messages increase non-linearly with the use of AUTOSAR SecOC profiles on the classical CAN bus—an 8 byte maximum payload—incurs untenable overhead for message frames with payloads greater than 3 bytes due to the need to fragment the message PDU plus the CMAC and FV into multiple frames. The baseline *payload* varies from 0 to 8 bytes with transmission time in a 1 Mbps CAN bus assuming maximum bit stuffing. (The fastest CAN bitrate is 1 Mbps with other common high-speed bitrates of 250 or 500 kbps.) Adding MACs increases the transmission time by at least the number of bits in the MAC and FV, but the real problem comes when additional data frames are needed to transmit the excess that does not fit in the 8-byte data payload. The frame overhead greatly increases the transmission time overhead incurred by the authentication. As a result, it is generally well understood that SecOC is not directly supportable for classical CAN, although it can be supported with the Controller Area Network Flexible Data-rate (CAN FD) and Controller Area Network extra long (CAN XL) payload protocols. Utilizing CAN FD messages to include MAC tags is also proposed in the work-in-progress SAE J1939-91a draft standard. However CAN remains a popular protocol, thus authentication schemes are proposed that aim to reduce the authentication overhead [6]–[8].

Due to the relatively large overhead of MACs compared to CAN data payloads, another approach besides (or in addition to) truncation is to transmit MACs periodically [9], [10]. Two variations on this approach are common, which we will refer to as skipping and batching. Skipping authenticates some messages in a sequence, i.e., authenticating one out of n messages. The batching approach authenticates a *batch* of several messages with one MAC, for example, Daily et al. [10] suggest authenticating all messages in a one second interval for CAN/J1939 traffic. In this batching scheme, messages were authenticated in batches based on the SAE J1939 Source Address, which is the last 8 bits in the extended CAN ID. This approach is sensitive to arrival order of the messages. These two approaches will incur different computational performance overhead for the cryptographic operations—skipping authentication means less work. The approach that skips messages only provides a probabilistic authentication, since those skipped messages will not have their integrity checked. The batched approach will still incur a similar (slightly smaller) cryptographic computational overhead while authenticating every message, but still ensures that every message's authenticity can be checked. Both skipping and batching however have the same benefit to the network overhead, because they both make MAC transmissions periodic. We treat these approaches identically and refer to them both as periodic MAC transmission. We ignore the fact that, prior to transmitting a message, software copies it into a buffer that the CAN controller will read from. Similarly, we do not discuss the need for the ECU to copy the message from its receive (RX) buffer before the MAC arrives and to hold it there until the MAC completes when utilizing the periodic MAC transmission. This overhead is separate from the RTA and is beyond the scope of our study.

Despite a plethora of authentication schemes for CAN, the prior work does not provide a sound, rigorous methodology to understand the real-time performance impact of adding authentication. In this paper, we provide a comprehensive approach to show how such impact can be quantified and understood based on the well-known theory of RTA for CAN schedulability. The contributions of this paper include:

- RTA formulation for CAN FD and CAN XL;
- RTA formulation for the addition of MACs to CAN, CAN FD, and CAN XL;
- RTA formulation for the use of periodic MACs with CAN, CAN FD, and CAN XL;
- and evaluation of SecOC authentication for CAN, CAN FD, and CAN XL, with case studies and randomized schedulability experiments.

II. Related Work

The network overhead introduced by cryptographic schemes for enhancing message authenticity in CAN—such as MaCAN [6] which employs shared keys between group of nodes—motivates deriving the impact authentication protocols have on the timely arrival of messages for automotive applications. Lin et al. [11] use MACs to protect against masquerade and replay attacks. However, they observed that adding MAC bits increases the message transmission time especially when they are fragmented over multiple messages. Fragmentation easily leads to violation of timing constraints and also affects system performance of CAN. In their efforts to reduce the negative impact on automotive electronic systems, the authors used mixed-integer linear programming-based technique (MILP) to assign priorities and allow multiple ECUs to share a MAC for design flexibility. Groza et al. [12] propose a different view on authentication that focuses on embedded authenticators for the CAN ID within the identifier field itself,

which may allow for faster detection of spoofed messages by authenticating the sender before the data payload transmission. Nürnberger et al. [13] proposed vatiCAN, a backward-compatible authentication mechanism based on a lightweight keyed-Hash Message Authentication Code (HMAC). The authors analyzed the authentication overhead and concluded that their implementation, which authenticates every single CAN frame, would result in an impractical bandwidth overhead. Van Bulck et al. [14] presented an authentication mechanism that allows multiple IDs to share the same key in order to save memory. They truncated the MAC by removing the eight least significant bytes and used hardware-level cryptography to achieve a reduction in MAC computation times.

Xie et al. [15] propose a security enhancement by adding MACs to messages in CAN FD taking advantage of the laxity interval from the lower bound to the deadline limited to parallel applications.

Other tangentially related work has examined the cryptographic overhead for securing CAN and CAN FD messages with respect to impact on the real-time performance of software tasks. Petit et al. [16] investigated the impact of elliptic curve digital signature algorithm as an authentication mechanism on braking distance in vehicular networks. They conducted analysis experiments to derive the total overhead in the packet size and when transmitting data in vehicle-to-vehicle communications. They highlighted the impact of authentication key size and proposed optimizations mechanisms to reduce the overhead. Wu et al. [17] analyze timing overhead from cryptography algorithms on Infineon's Aurix processors and find that software implementation of the AES-128 cryptographic algorithm cannot meet the timing constraints of distributed automotive cyber-physical systems. Hence, hardware acceleration is a common approach to mitigate the cryptographic overheads. To reduce the computational cost of using a hardware security module (HSM), Xie at al. [18] explore task assignment and message scheduling approaches to minimize the costs associated with security functions on CAN FD messages. Lesi et al. [19] also used MILP techniques to derive a schedulable task set for quality-of-control (QoC) requirements in the presence of attacks on sensor data. Their work provides analysis of the effect that computational overhead has on schedulability of control tasks and QoC in the software layer of embedded systems with application to an automotive case study. Ghosh et al. [20] propose a timed automata-based model for checking schedulability of control tasks augmented with reliability guaranteed by sensor fault mitigation techniques and security based on authentication techniques; however, their scheduling policy is limited to static scheduling where the order in which threads are executing is controlled manually.

Furthermore, Bordoloi and Samii [21] presented the frame packing problem for CAN FD and proposed an optimization approach for selecting and packing payloads into frames to minimize bus utilization. Their approach considers timing constraints and potential repacking of messages to satisfy deadlines and formulates the best- and worst-case transmission times of CAN FD. This formulation is flawed as the specification used was before ISO standardization which has been updated, and does not consider the 29-bit identifiers. Similarly, De Andrade et al. [22] presented an analytical worst-case response time analysis of messages in CAN FD. However, their analysis is based on the obsolete CAN transmission time presented by Tindell et al. [3], which has since been revised by Davis et al. [1].

Our work provides the first comprehensive approach to formulate RTA for CAN, CAN FD, and CAN XL when adding MACs and with the optimization of periodic MAC transmission. Our methodology can be extended for different authentication protocols and proposals.

III. Classical CAN Response Time Analysis

We adopt the widely used response time analysis for CAN devised by Tindell et al. [2], [3] and revised by Davis et al. [1]. The terminology and formulation of this model mimics that of task schedulability based on response time analysis with the replacement of tasks by messages and jobs by message instances. Table II summarizes our notation.

The priority of each message is defined by its identifier (ID), which is a unique integer of 11 bits with base frame formats or 29 bits when using the extended CAN frame format. Lower numeric ID values indicate higher priority. Messages may be periodic (time-triggered) or sporadic (event-triggered). The former release message instances every *period* time unit and the latter release message instances separated by at least some *minimum inter-arrival time*. Each instance is subject to *jitter* that varies based on computational overheads, queueing delays, network availability, and physical effects in the communication medium.

Without loss of generality, we assume that a sequence of n messages in a set M are ordered as M_1 through M_n with the ID of any message M_i equal to i. A message M_i's worst-case response time (WCRT) is the maximum of its instances' response times in a *level-i busy interval*, which is a span of message transmissions with priorities greater than or equal to i. The busy interval ends when an instance transmits with priority less than i, or when the bus is idle. M_i's WCRT is given by

$$R_i = \max_{q \in [0, Q_i - 1]} R_i(q) \tag{1}$$

where Q_i is a count of instances for M_i that release during the busy interval. A message instance q in the busy interval has a WCRT of $R_i(q)$. These variables are found by solving

$$R_i(q) = J_i + w_i(q) - qP_i + C_{D_i} \tag{2}$$

$$Q_i = \left\lceil \frac{t_i + J_i}{P_i} \right\rceil \tag{3}$$

The (queueing) jitter of the frame given as J_i providing an upper bound on the time between when an instance releases and when it is ready, i.e., attempting to arbitrate for bus access. P_i is the period (or minimum inter-arrival time) of M_i. The maximum transmission time of q is given by C_{D_i}.

The contents of a message instance determine its transmission time, which is an integer multiple of τ_{bit}, the transmission time of a single bit. A message instance has a data payload between 0 and 8 bytes in length, which is given in the instance's

TABLE II: Table of Notation for Response Time Analysis

Variable	Definition
M	set of messages $M = (M_1, M_2, \ldots, M_n)$
$M_i \in M$	the ith message
D_i	message i's payload length or DLC
C_x	transmission time with x-byte payload
P_i	message period/inter-arrival time
R_i	worst case response time
J_i	queuing jitter
w_i	queuing delay
B_i	blocking time
I_i	interference time
τ_{bit}	the transmission time of a single bit
τ_{dbit}	the transmission time of a data bit (CAN FD)
E_i	the error overhead
β_i, α_i	authentication frames for M_i

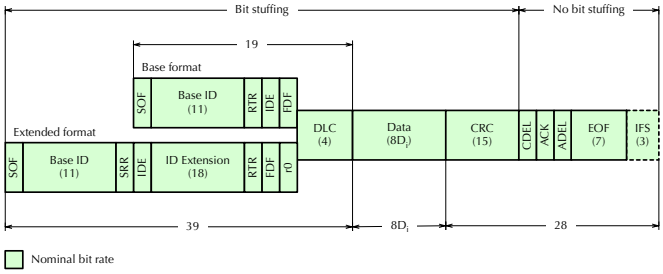


Fig. 2: Classical CAN Frame Format (Stuff Bits Not Shown)

Data Length Code (DLC). The worst-case transmission time C_{D_i} of a message M_i depends only on the frame format (base or extended) and on the payload length D_i in bytes, i.e., the value of the DLC. C_{D_i} can be calculated considering the worst possible bit stuffing pattern, that is, 5 dominant bits (including the SOF, to trigger the injection of the first stuff bit) followed by an alternating pattern made up of groups of 4 recessive and 4 dominant bits [23], [24]. Then for the base frame format the worst-case transmission time is given by

$$C_{D_i}^{(C,b)} = (55 + 10D_i)\tau_{bit}, \tag{4}$$

where the superscript (C, b) denotes the classical base frame format. For RTA, this value includes the three mandatory intermission frame space (IFS) bits that act as a spacer between frames, although technically they are not part of the CAN frame itself. As shown in Figure 2, using the extended instead of the base frame format adds 20 bits to the best-case (no stuff bits) frame length and up to 5 more stuff bits, so the worst-case transmission time becomes

$$C_{D_i}^{(C,e)} = (80 + 10D_i)\tau_{bit} \tag{5}$$

where the superscript (C, e) denotes the classical extended frame format.

Depending on the frame format of a message instance M_i the value of C_{D_i} will take on either Eq. 4 or Eq. 5. It is worth noting that the theoretical worst-case bit stuffing pattern mentioned previously cannot occur in practice due at least to the fixed-form bits in the CAN frame which necessarily break it. Therefore in both base and extended frame cases the formula for C_{D_i} gives a slightly pessimistic upper bound.

An upper bound on how long a message instance can spend waiting to transmit is $w_i(q)$ and the worst-case delay w_i, given an error model is found by solving the recurrence

$$w_i^{n+1}(q) = B_i + E(w_i^n + C_{D_i}) + qC_{D_i} + I_i \tag{6}$$

starting with $w_i^0(q) = B_i + qC_{D_i}$ and terminating at $w_i^{n+1}(q) = w_i^n(q)$. Where B_i is the maximum *blocking* time caused by lower-priority messages already transmitting on the bus when q is ready,

$$B_i = \max_{k>i}(C_{D_k}), \tag{7}$$

and I_i for *interference* caused by higher-priority messages that beat q during arbitration:

$$I_i = \sum_{k<i} \left\lceil \frac{w_i + J_k + \tau_{bit}}{P_k} \right\rceil C_{D_k}. \tag{8}$$

τ_{bit} is the time needed to transmit a single bit, which is the multiplicative inverse of the bus bitrate.

The length of the level-i busy interval is given by t_i, and it is found by solving the recurrence relation

$$t_i^{n+1} = B_i + E_i(t_i^n) + \sum_{k\leq i} \left\lceil \frac{t_i^n + J_k}{P_k} \right\rceil C_{D_k} \tag{9}$$

starting with $t_i^0 = C_{D_i}$ and terminating at $t_i^{n+1} = t_i^n$.

Message error handling, E_i is formulated as

$$E_i(t_i) = \Big(31\tau_{bit} + \max_{k\geq i}(C_{D_k})\Big)F(t_i) \tag{10}$$

with 31 bits for the error signal. $F(t_i)$ is a step function defined by a fault model to determine an upper-bound on errors over a time interval. It must be a monotonic non-decreasing function. Broster et al. [25] suggest a harsh environment may induce an expected number of errors at 30 faults per second.

A message is said to be *schedulable* if its WCRT is less than or equal to its *deadline*. Message deadlines may be *implicit*, i.e., equal to their period, or they may be *explicit* (constrained). Explicit deadlines are often used for safety-critical messages that may have a long period, e.g., 500 ms, but require a response much sooner.

IV. CAN Flexible Data-Rate (FD) Response Time Analysis

The Controller Area Network Flexible Data-Rate (CAN FD) protocol extends classical CAN primarily with the ability to accommodate larger data payloads and to switch to a higher bitrate for portions of the frame transmission. CAN FD frames accommodate up to 64 data bytes. The payload length however is still encoded as a 4-bit DLC value, so only a few predefined lengths are available. For this reason, we introduce a function $z(D_i)$ that gives the smallest feasible payload length able to accommodate D_i bytes. The function is monotonic and is defined as:

$$z(D_i) = \begin{cases} D_i & (0 \leq D_i \leq 8) \\ 12 & (8 < D_i \leq 12) \\ 16 & (12 < D_i \leq 16) \\ 20 & (16 < D_i \leq 20) \\ 24 & (20 < D_i \leq 24) \\ 32 & (24 < D_i \leq 32) \\ 48 & (32 < D_i \leq 48) \\ 64 & (48 < D_i \leq 64) \end{cases}. \tag{11}$$

TABLE III: Table of Notation, CAN FD Bit Stuffing

Part of the frame	Length	Stuff bits
Bits between SOF and BRS, included	n_b	s_b
Bits after BRS and before the CRC field	n_d	s_d
Bits in the CRC field	n_f	s_f
Bits after CDEL	n_u	s_u

TABLE IV: Variable Values, CAN FD Bit Stuffing

Var.	Base Frame Format	Extended Frame Format	P. length $D_i \leq 16$	P. length $D_i > 16$
n_b	$n_b^{(b)} = 17$	$n_b^{(e)} = 36$	(independent of D_i)	
n_d	$n_d = 5 + 8z(D_i)$		(monotonic in D_i)	
n_f	(independent of format)		$n_f^{(D_i \leq 16)} = 22$	$n_f^{(D_i > 16)} = 26$
n_u	$n_u = 13$			

In addition, CAN FD optionally supports a bitrate switch (BRS) to a higher bitrate for part of the frame, where the bit transmission time becomes $\tau_{dbit} \leq \tau_{bit}$.

Worst-case frame length calculation becomes more complex in CAN FD than in classical CAN because of two factors. Firstly, CAN FD replaces the on-demand bit stuffing of classical CAN with *fixed bit stuffing* in the CRC field (stuff count, CRC sequence, and CDEL). There, a stuff bit is unconditionally injected every four CRC field bits, regardless of bit stream contents. Secondly, part of the frame and any stuff bits it contains may be transmitted at a higher bitrate (bit transmission time τ_{dbit} instead of τ_{bit}) due to BRS.

As shown in Figure 3 and summarized in Table III, a CAN FD frame can be divided into four parts for worst-case C_{D_i} calculation:

1) n_b bits between the SOF and BRS bit, transmitted with on-demand bit stuffing at the nominal bitrate, except for the BRS bit where bitrate switching (to the data bitrate) may take place.
2) n_d bits transmitted with on-demand bit stuffing at the data bitrate after BRS and before the CRC field.
3) n_f bits in the CRC field, transmitted with fixed bit stuffing at the data bitrate, except for the CDEL bit (the last bit of the CRC field) where bitrate switching (back to the nominal bitrate) may take place.
4) n_u bits after the CRC field, transmitted without bit stuffing at the nominal bitrate. In CAN FD this part of the frame may consist of up to two ACK slots, instead of one like in classical CAN. This is because the standard [26, Section 10.4.2.7] specifies that nodes shall accept overlapping ACKs, sent by different receivers, which are up to two bits long overall. This leads to the value of n_u we consider in schedulability analysis. The ACK slot(s) are followed by the 1-bit ACK delimiter, and the 7-bit end of frame (EOF).

As before, for RTA n_u includes the three mandatory IFS bits. Table IV gives the values of the above variables depending on the frame format (base or extended) and the payload length. $n = n_b + n_d + n_f + n_u$ is the total number of unstuffed bits in the frame. The worst-case number of stuff bits s_b needed by the first n_b bits of a CAN FD frame can be calculated considering the worst-case pattern for on-demand bit stuffing, which in CAN FD is still the same as for classical CAN. We obtain:

$$s_b = \left\lfloor \frac{n_b - 1 - 1}{4} \right\rfloor. \tag{12}$$

The two corrective terms -1 in the numerator of (12) are needed to take into account that the primer of the worst-case bit stuffing pattern consists of 5 bits instead of 4, and to avoid counting in s_b a stuff bit to be injected immediately after BRS, because that stuff bit is transmitted at the data rather than the nominal bitrate.

The number of unstuffed bits at the same value up to and including BRS after the last stuff bit transmitted at the nominal bitrate is given by:

$$r_b = (n_b - 1) - 4s_b. \tag{13}$$

A value $r_b = 4$ indicates that a stuff bit must be transmitted, at the data bitrate, immediately after BRS. The worst-case number of stuff bits s_d needed by the next n_d bits is then given by:

$$s_d = \left\lfloor \frac{r_b + n_d - 1}{4} \right\rfloor. \tag{14}$$

In (14) the -1 at the numerator avoids counting an on-demand stuff bit to be injected immediately before the CRC field. This is because there is always a fixed stuff bit at the very beginning of the CRC field and the standard specifies that no two consecutive stuff bits may appear in the frame.

Finally, the number of fixed stuff bits s_f needed by the n_f bits in the CRC field is:

$$s_f = \left\lceil \frac{n_f}{4} \right\rceil. \tag{15}$$

In this equation, we use a ceiling instead of a floor operator to count the stuff bit injected before the first bit of the CRC field. Due to the allowed values of n_f no stuff bit is ever required immediately after CDEL. All subsequent n_u bits up to the end of the frame are transmitted without bit stuffing, so it (trivially) is $s_u = 0$.

The total, worst-case transmission time C_{D_i} of a message M_i can be expressed as the sum of the contributions of the four parts of the frame just described, that is

$$C_{D_i}^{(F)} = (n_b + s_b + n_u)\tau_{bit} + (n_d + s_d + n_f + s_f)\tau_{dbit} \tag{16}$$

where the superscript (F) is used to denote CAN FD.

Equation (16) considers the BRS and CDEL bits to be entirely transmitted at the nominal and data bit rate, respectively, although this is not strictly true because in those bits BRS takes place. The equation is still correct because the CAN standard [26, p. 46] stipulates that "the sum of the length of these two bits is the same as the sum of one bit of the nominal bit time and one bit of the data bit time."

Substituting the values of Table IV and (12–15) into (16) four cases are possible, depending on the frame format and payload length D_i, in bytes:

$$\begin{aligned} C_{D_i}^{(F,b,D_i \leq 16)} &= 33\tau_{bit} + (35 + 10z(D_i))\tau_{dbit} \\ C_{D_i}^{(F,b,D_i > 16)} &= 33\tau_{bit} + (40 + 10z(D_i))\tau_{dbit} \\ C_{D_i}^{(F,e,D_i \leq 16)} &= 57\tau_{bit} + (34 + 10z(D_i))\tau_{dbit} \\ C_{D_i}^{(F,e,D_i > 16)} &= 57\tau_{bit} + (39 + 10z(D_i))\tau_{dbit} \end{aligned} \tag{17}$$

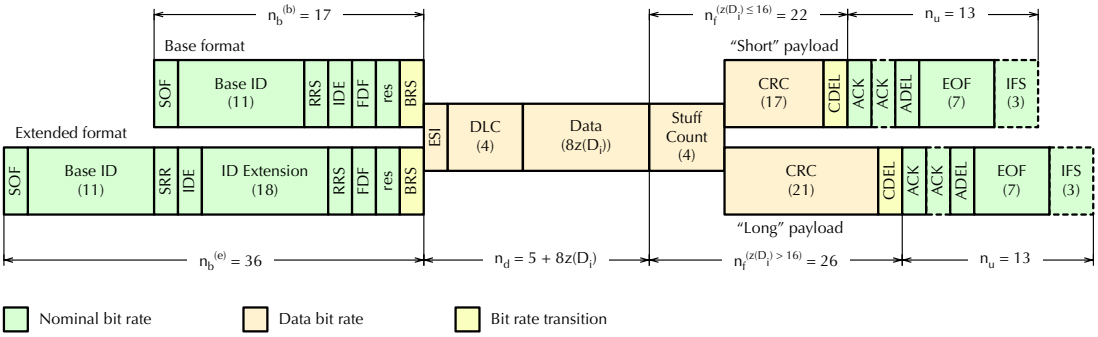


Fig. 3: CAN FD Frame Format (Stuff Bits Not Shown)

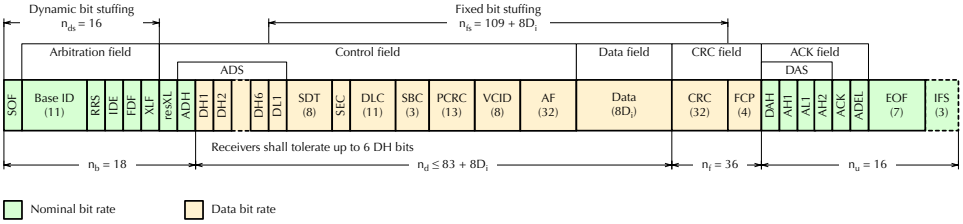


Fig. 4: CAN XL Frame Format (Stuff Bits Not Shown)

TABLE V: Table of Notation, CAN XL Bits

Var	Value	Description
n_b	18	Bits from SOF to ADH included
n_d	$83 + 8D_i$	Bits between ADH and the CRC field
n_f	36	Bits in the CRC field
n_u	16	Bits after the CRC field
n_{ds}	16	Bits with dynamic bit stuffing
n_{fs}	$109 + 8D_i$	Bits with fixed bit stuffing

Here the superscript indicates the frame format (F, b for base and F, e for extended) and the size of the data payload below (inclusive) or above 16 bytes. The remainder of the RTA for CAN FD is unmodified, i.e., the WCRT are solved for Eq. 1 selecting the equation for C_{D_i} from Eq. 17 based on the frame format and the value of D_i.

V. Controller Area Network Extra Long (CAN XL) Response Time Analysis

The Controller Area Network extra long payload (CAN XL) extends the CAN and CAN FD protocol with much larger data payload. CAN XL is currently not standardized and our analysis is based on a draft standard that remains subject to change [27]. Similar to CAN FD, CAN XL supports bitrate switching and data frame transfer with two bit rates. Bitrate switching is required in CAN XL, and extended frame formats are not supported (no 29-bit identifier). In the arbitration phase, a maximum of 1Mbit/s is allowed with a limited payload length, similar to the classical CAN bit rate requirement. However in the data phase, the bitrate can be 10Mbit/s or higher while the bit rate is not limited by the network length. With CAN XL accommodating up to 2048 bytes of data, Ethernet frames can be packed in CAN XL messages.

CAN XL adds more overhead than CAN FD in terms of the fixed stuff bits in the data phase of the frame and dynamic bit stuffing in the header CRC as a stuff bit is added after 10 data bits.

As shown in Figure 4, a CAN XL frame can be divided into several parts for frame length calculation. Table V summarizes the notation used in the description that follows.

1) The first part of the frame consists of n_b bits transmitted at the nominal bit rate. They comprise the SOF bit, the arbitration field, as well as the resXL bit and the ADH bit of the Arbitration to Data Sequence (ADS) at the beginning of the control field.
2) It is followed by n_d bits transmitted at the data bit rate, encompassing the remainder of the control field and the data field. n_d depends linearly on D_i.
3) The CRC field immediately follows the data field and consists of n_f bits.
4) The last part of a CAN XL frame is very similar to classical and FD frames; it consists of n_u bits.

Although there are only two DH bits in the control field of a nominal CAN XL frame, we considered six in frame length calculation. This is because the draft standard [27, Section 6.8.5.4] specifies that CAN XL nodes shall tolerate a minimum of one and a maximum of 6 DH bits without flagging a form error, with the purpose of nullifying the phase error due to the change of transceiver operating mode and bit encoding method. Unlike in CAN FD, CAN XL bit rate transitions take place at bit boundaries rather than at the sampling point of BRS and CDEL. As a consequence, in CAN XL there are no bits transmitted partly at the nominal and partly at the data bit rate.

The worst-case frame length of a CAN XL frame carrying a payload of D_i bytes without taking bit stuffing into account is therefore:

$$(n_b + n_u)\tau_{bit} + (n_d + n_f)\tau_{dbit}. \tag{18}$$

Bit stuffing is performed in two different ways in two non-overlapping regions of the frame, while the rest is not bit stuffed. Most importantly, none of these regions contain a bit

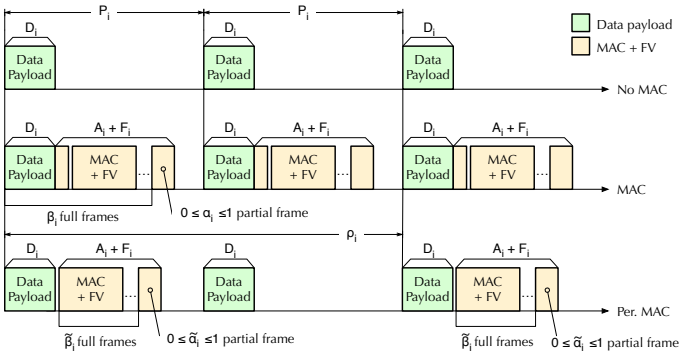


Fig. 5: Comparison of frame sequences for message M_i with No MAC, MAC and Periodic MAC ($\rho_i = 2P_i$) transmission. (Frame header, CRC, and trailer not shown; no interference/blocking by other messages.)

rate transition, which makes calculations simpler than for CAN FD. In particular:

1) The first n_{ds} bits of the frame (SOF plus arbitration field) are bit stuffed with *dynamic* bit stuffing exactly as in classical CAN. The worst-case number of stuff bits needed in this region is:

$$s_{ds} = \left\lfloor \frac{n_{ds} - 1}{4} \right\rfloor = 3, \tag{19}$$

where the corrective term -1 in the numerator is needed to take into account that the primer of the worst-case bit stuffing pattern consists of 5 bits instead of 4. Dynamic stuff bits are transmitted at the nominal bit rate.

2) The n_{fs} bits starting at DL1 and up to the end of the CRC in the CRC field are subject to *fixed* bit stuffing, in which one stuff bit at the opposite polarity of the previous one is unconditionally inserted every 10 bits. The number of stuff bits in this region is then:

$$s_{fs} = \left\lfloor \frac{n_{fs}}{10} \right\rfloor = \left\lfloor \frac{109 + 8D_i}{10} \right\rfloor = 10 + \left\lfloor \frac{9 + 8D_i}{10} \right\rfloor. \tag{20}$$

Unlike in (19), there is no corrective term because no primer sequence is needed to initiate fixed bit stuffing. Fixed stuff bits are transmitted at the data bit rate.

By combining (18)–(20) we can finally express the worst-case transmission time $C_{D_i}^{(X)}$ of a CAN XL message M_i as:

$$\begin{aligned} C_{D_i}^{(X)} &= (n_b + n_u + s_{ds})\tau_{bit} + (n_d + n_f + s_{fs})\tau_{dbit} \\ &= (18 + 16 + 3)\tau_{bit} \\ &+ \left(83 + 8D_i + 36 + 10 + \left\lfloor \frac{9 + 8D_i}{10} \right\rfloor\right)\tau_{dbit} \\ &= 37\tau_{bit} + \left(129 + 8D_i + \left\lfloor \frac{9 + 8D_i}{10} \right\rfloor\right)\tau_{dbit}. \end{aligned} \tag{21}$$

Using this formulation from Eq. 21, the rest of the RTA for CAN XL proceeds as in the original CAN with WCRT solved for Eq. 1.

VI. MAC Response Time Analysis

Cryptographic approaches for authentication append a message authentication code (MAC) to transmitted data. A minimum of 64 bits is considered necessary to provide some level of protection from successful spoofing attacks [28]. In automotive applications, full MACs are therefore typically 8 or 16 bytes long, and most authentication protocols furthermore require the transmission of freshness values (FVs) to prevent replay attacks. CAN data frames can only carry up to an 8-byte payload, and therefore it cannot append a full MAC with FV to any transmitted data. As a result, the state-of-the-art approaches provide weakened forms of authentication via truncated, batched, or periodically transmitted MACs. In the following we examine the addition of MACs to CAN with respect to its response time analysis.

Including MAC overhead in the CAN response time presents two key challenges: (1) the generation of additional message frames is needed for each time that the data payload exceeds the maximum frame's payload length; (2) these additional message frames are generated at the same time as the message instance, which slightly changes the periodic message model to accommodate for multiple instances per release. Interestingly, some vehicle manufacturers do generate multiple periodic message instances of the same ID, which has not been considered by the prior work in CAN response time analysis. To address the first challenge, we begin by formulating β_i as the number of full message frames required for transmitting authentication information appended to the original data payload of D_i bytes as a MAC of length A_i and FV of length F_i in bytes, as shown in Fig. 5. Hence,

$$\beta_i = \left\lfloor \frac{D_i + A_i + F_i}{D_{\max}} \right\rfloor, \tag{22}$$

where $D_{\max}$ is the protocol's maximum data payload size in byte, which is 8 in CAN for both base and extended frame formats. Note that β_i may be zero in case the message and authentication overhead is smaller than $D_{\max}$, and it may be greater than one in case the message and authentication overhead are greater than or equal to $2 * D_{\max}$. We also formulate α_i as the number (at most one) of partial message frames:

$$\alpha_i = \left\lceil \frac{D_i + A_i + F_i}{D_{\max}} \right\rceil - \beta_i. \tag{23}$$

This formulation allows for the possibility of specifying different maximum sizes for each message's data payload and authentication information.

When the original data payload and the authentication overhead cause fragmentation to occur, i.e., $\beta_i + \alpha_i > 1$, the extra message instances (or fragments) can have differing worst-case transmission times—but the same deadline—that need to be reflected in the calculation of the busy interval and response time. An important insight here is that the aggregate sequence of fragments can be considered as one instance taken together, because only the first fragment encounters

blocking. Thus, we find the worst-case transmission time for the sequence of the $\beta_i + \alpha_i$ message fragments as

$$\widehat{C_{D_i}} = \beta_i C_{D_{\max}} + \alpha_i C_{(D_i+A_i+F_i) \mod D_{\max}}. \quad (24)$$

Note that α_i is redundant here as we calculate a non-zero worst-case transmission time in the latter term only when $D_i + A_i + F_i$ is not an exact multiple of $D_{\max}$. If $D_i + A_i + F_i$ is a multiple of $D_{\max}$ then the transmission times will be accounted for by multiplication of the first term by β_i.

Similarly, the blocking term induced by lower-priority messages may change due to the inclusion of authentication information in the same frame as the original data payload, hence

$$\widehat{B_i} = \max_{k>i}(\max(\lceil \beta_k/(\beta_k+1) \rceil C_{D_{\max}}, C_{(D_k+A_k+F_k) \mod D_{\max}})). \quad (25)$$

Here the ceiling of division of β_k by $\beta_k + 1$ ensures that if $\beta_k > 0$ then one $C_{D_{\max}}$ is included as the blocking term induced by M_k, while if $\beta_k = 0$ then only the partial frame for M_k is included.

The additional message frames then modify Q_i from Eq. 3 to

$$\widehat{Q_i} = \left\lceil \frac{\widehat{t_i} + J_i}{P_i} \right\rceil (\beta_i + \alpha_i). \quad (26)$$

The multiplication by $(\beta_i + \alpha_i)$ accounts for the number of frames transmitted for each periodic (or sporadic) instance of M_i, while the term within the ceiling remains as in Eq. 3.

Due to the extra instances generated, the level-i busy interval from Eq. 9 changes to $\widehat{t_i}$, and it is found by solving the recurrence relation

$$\widehat{t_i^{n+1}} = \widehat{B_i} + E_i(\widehat{t_i^n}) + \sum_{k \le i} \left\lceil \frac{\widehat{t_i^n} + J_k}{P_k} \right\rceil \widehat{C_{D_k}} \quad (27)$$

starting with $\widehat{t_i^0} = \widehat{C_{D_i}}$ and terminating at $\widehat{t_i^{n+1}} = \widehat{t_i^n}$.

The second challenge requires modification of the WCRT $R_i(q)$ of message instance q (Eq. 2). Here we need to adjust the waiting time $w_i(q)$ from the recurrence in Eq. 6 to include the interference due to the extra message instances, so the new recurrence becomes

$$\begin{aligned}\widehat{w_i^{n+1}}(q) = {} & \widehat{B_i} + E(\widehat{w_i^n} + \widehat{C_{D_i}}) \\ & + (q \mod (\beta_i + \alpha_i)) C_{D_{\max}} \\ & + \left\lfloor \frac{q}{\beta_i + \alpha_i} \right\rfloor \widehat{C_{D_i}} + \widehat{I_i}\end{aligned} \quad (28)$$

where

$$\widehat{I_i} = \sum_{k<i} \left\lceil \frac{\widehat{w_i} + J_k + \tau_{bit}}{P_k} \right\rceil \widehat{C_{D_k}}. \quad (29)$$

The third term in Eq. 28 of $(q \mod (\beta_i + \alpha_i))$ captures the number of full message frames previously transmitted as fragments of the same message instance, then multiplied by the maximum transmission time of $C_{D_{\max}}$. This term accounts for the delay on later fragments caused by earlier fragments released for the same message instance. Similarly, the fourth term of $\lfloor q/(\beta_i + \alpha_i) \rfloor$ represents the number of previously transmitted (possibly fragmented) message instances for the same message M_i within the i-level busy interval.

The recurrence is solved starting with $\widehat{w_i^0}(q) = \widehat{B_i} + \lfloor \frac{q}{\beta_i+\alpha_i} \rfloor \widehat{C_{D_i}}$ and terminating at $\widehat{w_i^{n+1}}(q) = \widehat{w_i^n}(q)$. This formulation avoids the pessimism that would result from including $\widehat{B_i}$ in every fragment of the same instance because only the first fragment can incur blocking.

The WCRT of the instance q is adjusted to change the subtraction of its offset within the busy interval to account for the multiple simultaneous release of fragments in each period, and to add either $C_{D_{\max}}$ or $(C_{(D_i+A_i+F_i) \mod D_{\max}})$ depending on whether q is for an earlier or final fragment of a message instance, respectively, hence

$$\begin{aligned}\widehat{R_i}(q) = {} & J_i + \widehat{w_i(q)} - \left\lfloor \frac{q}{\beta_i + \alpha_i} \right\rfloor P_i \\ & + \left(1 - \left\lfloor \frac{(q \mod (\beta_i + \alpha_i)) + 1}{\beta_i + \alpha_i} \right\rfloor \right) C_{D_{\max}} \\ & + \left\lfloor \frac{(q \mod (\beta_i + \alpha_i)) + 1}{\beta_i + \alpha_i} \right\rfloor \Gamma_i\end{aligned} \quad (30)$$

where

$$\begin{aligned}\Gamma_i = {} & \alpha_i (C_{(D_i+A_i+F_i) \mod D_{\max}}) \\ & + ((1-\alpha_i)(\beta_i - 1) C_{D_{\max}}).\end{aligned} \quad (31)$$

Here we again have $\lfloor q/(\beta_i + \alpha_i) \rfloor$ in the third term as the number of complete transmitted message instances prior to q, which is multipled by P_i to determine the offset within the i-level busy interval of the current message instance. The fourth and fifth terms both include $\lfloor ((q \mod (\beta_i + \alpha_i)) + 1)/(\beta_i + \alpha_i) \rfloor$, which yields a value of 1 for the final fragment in a message instance and 0 for fragments of an incomplete message instance. In the fifth term we introduce Γ_i for presentation only, with Eq. 31 yielding either the transmission time of a partial message in case $\alpha_i = 1$ or yielding $C_{D_{\max}}$ in case $\alpha_i = 0$ and $\beta_i > 1$. We generally expect that $0 \le \beta_i \le 2$, but even for larger values of β_i the worst-case response time of the last fragment in a message instance upper-bounds the other fragments, which is most likely all that matters in practice since all fragments must arrive at the destination within the deadline to be useful.

Finally we have

$$\widehat{R_i} = \max_{q \in [0, \widehat{Q_i} - 1]} \widehat{R_i}(q) \quad (32)$$

as the response time with MAC overhead.

VII. Periodic MAC Response Time Analysis

The primary change needed in the CAN RTA to accommodate periodic MACs is to generate additional message frames that will carry the MAC at a periodic rate ρ_i. We do not constrain the size of the MAC but we would anticipate that the MAC would fit in one full data frame. We assume the rate is a multiple of the message's period P_i, i.e., they are harmonic. Also, when message frames are sporadic, the MAC is sent every ρ_i/P_i transmissions of message M_i. Furthermore, we assume that the periodic MAC is transmitted separately (albeit

with the same ID) from data payloads; this assumption could be relaxed but the following formulation would become more complicated.

Hence, we first modify Eq. 22 and Eq. 23 to determine the number of full frames and (at most one) partial frames needed for authentication as

$$\widetilde{\beta}_i = \left\lfloor \frac{A_i + F_i}{D_{\max}} \right\rfloor \tag{33}$$

$$\widetilde{\alpha}_i = \left\lceil \frac{A_i + F_i}{D_{\max}} \right\rceil - \widetilde{\beta}_i \tag{34}$$

and the modification of Q_i from Eq. 3 ($\widehat{Q_i}$ from Eq. 26 respectively) becomes

$$\widetilde{Q_i} = \left\lceil \frac{\widetilde{t_i} + J_i}{P_i} \right\rceil + \left\lceil \frac{\widetilde{t_i} + J_i}{\rho_i} \right\rceil (\widetilde{\beta}_i + \widetilde{\alpha}_i). \tag{35}$$

Here we include the cost of authentication in terms of the number of frames transmitted each interval of ρ_i. Separation of the authenticator from the data payload means that C_{D_i} remains valid for data message transmissions while the worst-case transmission time of the authenticator is

$$\widetilde{C_{D_i}} = \widetilde{\beta}_i C_{D_{\max}} + \widetilde{\alpha}_i C_{(A_i+F_i) \mod D_{\max}}. \tag{36}$$

and the blocking time is

$$\widetilde{B_i} = \max_{k>i}(\max(C_{D_k}, \lceil \beta_k/(\beta_k+1) \rceil C_{D_{\max}}, C_{(A_k+F_k) \mod D_{\max}})). \tag{37}$$

The level-i busy interval is now $\widetilde{t_i}$ found by solving the recurrence relation

$$\widetilde{t_i^{n+1}} = \widetilde{B_i} + E_i(\widetilde{t_i^n}) + \sum_{k \le i} \left(\left\lceil \frac{\widetilde{t_i^n} + J_k}{P_k} \right\rceil C_{D_k} + \left\lceil \frac{\widetilde{t_i^n} + J_k}{\rho_k} \right\rceil \widetilde{C_{D_k}} \right) \tag{38}$$

starting with $\widetilde{t_i^0} = C_{D_i} + \widetilde{C_{D_i}}$ and terminating at $\widetilde{t_i^{n+1}} = \widetilde{t_i^n}$.

We again need to adjust the waiting time $\widehat{w_i}(q)$ from the recurrence in Eq. 28 to include the interference due to the periodic generation of extra instances, hence

$$\begin{aligned} \widetilde{w_i^{n+1}}(q) = {} & \widetilde{B_i} + E(\widetilde{w_i^n} + C_{D_i}) \\ & + \left\lfloor \frac{q}{\rho_i/P_i + (\widetilde{\beta}_i + \widetilde{\alpha}_i)} \right\rfloor \left(\frac{\rho_i}{P_i} C_{D_i} + \widetilde{C_{D_i}} \right) \\ & + \min(q', \frac{\rho_i}{P_i}) C_{D_i} + \max(0, q' - \frac{\rho_i}{P_i}) C_{D_{max}} \\ & + \widetilde{I_i} \end{aligned} \tag{39}$$

where

$$q' = q \mod (\rho_i/P_i + (\widetilde{\beta}_i + \widetilde{\alpha}_i)) \tag{40}$$

and

$$\widetilde{I_i} = \sum_{k<i} \left(\left\lceil \frac{\widetilde{w_i} + J_k + \tau_{bit}}{P_k} \right\rceil C_{D_k} + \left\lceil \frac{\widetilde{w_i} + J_k + \tau_{bit}}{\rho_k} \right\rceil \widetilde{C_{D_k}} \right). \tag{41}$$

$\widetilde{I_i}$ is the interference caused by the regular message instance transmissions each P_k plus the interference caused by the periodic authentication each ρ_k. Here we calculate on one hand $\lfloor q/(\rho_i/P_i + (\widetilde{\beta}_i + \widetilde{\alpha}_i)) \rfloor$ the count of periodic authentications in the interval leading up to but not including the same batch of periodically authenticated message instances as q. Note that ρ_i/P_i yields the number of message instances transmitted between authentications, and therefore $\rho_i/P_i + (\widetilde{\beta}_i + \widetilde{\alpha}_i)$ is a count of instances transmitted when including the authentication message instance(s). We then multiply this count by the time needed to transmit all the messages in the batch. This corresponds to the third item in Eq. 39. On the other hand, $q' = q \mod (\rho_i/P_i + (\widetilde{\beta}_i + \widetilde{\alpha}_i))$ counts the number of messages (including the regular message instances and the authentication message instance(s) transmitted before instance q within the last batch. If $q' < \rho_i/P_i$, instance q corresponds to a regular message instance, otherwise it corresponds to the authentication message instance if $q' \ge \rho_i/P_i$. The fourth and fifth terms in Eq. 39 calculate the delay introduced due to messages transmitted before instance q within the same batch, which include regular message instances and optionally authentication message instance(s) depending on instance q.

The recurrence is solved starting with $\widetilde{w_i^0}(q) = \widetilde{B_i} + qC_{D_i}$ and terminating at $\widetilde{w_i^{n+1}}(q) = \widetilde{w_i^n}(q)$.

We also need to adjust the WCRT of an instance q in the busy period by subtracting its offset based on the periodic rate of MACs, hence

$$\widetilde{R_i}(q) = J_i + \widetilde{w_i}(q) - \left\lfloor \frac{q}{\rho_i/P_i + \widetilde{\beta}_i + \widetilde{\alpha}_i} \right\rfloor \rho_i + \max(C_{D_i}, \widetilde{C_{D_i}}). \tag{42}$$

Here, the floor function in $\lfloor q/(\rho_i/P_i + (\widetilde{\beta}_i + \widetilde{\alpha}_i) \rfloor$ gives us a lower bound of the periodic authentications before the batch containing q. We subtract another ρ_i as the offset each time that q exceeds a multiple of this count. Unlike the case of authenticating every instance, in the periodic case we do not know which values of q represent MACs, so we have to be a bit pessimistic in terms of potentially undercounting the number of periodic authentications prior to q as well as adding the worst-case transmission time of the larger of the data and its authenticator.

Finally, we have

$$\widetilde{R_i} = \max_{q \in [0, \widetilde{Q_i} - 1]} \widetilde{R_i}(q) \tag{43}$$

as the response time with periodic MACs.

VIII. Evaluation

A. Classical CAN

To analyze the formulation and performance of RTA for CAN, we evaluate the real-time performance impact of adding message authentication using synthetic workloads generated randomly and two uses cases: a realistic instrument cluster message set from the BMW E90 described by Buttigieg et al. [29] and an SAE benchmark described by Tindell and Burns [30]. Python implementations for all our experiments

are available as open-source[1]. The two use cases demonstrate the impact of MACs and periodic MACs on the response time of realistic CAN message sets. The synthetic workloads are used to conduct a randomized schedulability experiment with parameters that are representative of a wide variety of possible CAN message sets.

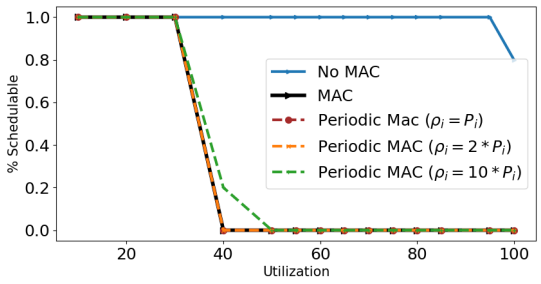
1) Use Case: BMW Message Set: We used the BMW CAN message set that depicts message signals sent in the instrument cluster of the BMW E90 car with a bus speed of 100 kb/s [29]. Table VI highlights the message IDs, DLC, periods, and their respective transmission times used for the experiment. We applied the RTA formulations for the message set as-is, and the addition of MACs and periodic MACs using SecOC Profile 1. The results obtained for the WCRT of the message IDs are also given in Table VI.

The response time of messages increased substantially when the MAC and the periodic MAC are added. However, increasing ρ_i greatly improves WCRTs. We also show the WCRT of the periodic MAC approach when $\rho_i = P_i$, i.e., when every message instance has a MAC applied. Functionally this approach is similar to the MAC approach with the exception that authenticators are not packed into the same frame as the original message payload data. Therefore, this periodic approach should always have WCRT greater than or equal to the MAC approach, which is shown in the results and lends confidence that our RTA formulations for the two approaches are consistent. The MAC approach is schedulable, although we calculated $\widehat{R_{0CE}} = 9.4$ so there is not much slack. The periodic MAC approach using $\rho_i = 2P_i$ is schedulable and has response times lower than the MAC approach, thus demonstrating the value of both our RTA formulation as well as periodic MACs for classical CAN.

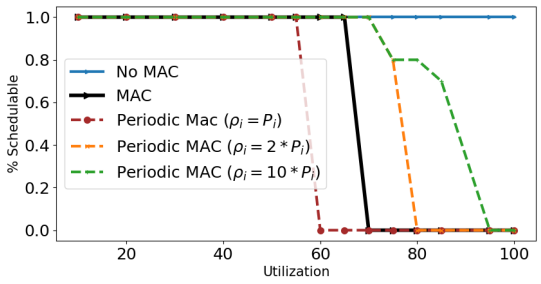
2) Use Case: SAE Benchmark: Table VII shows the parameters of the SAE benchmark, which is a modified version of the original SAE benchmark [30], along with the response time of messages with the MAC and periodic MAC. As can be seen at bus speed 125 kb/s, adding MACs and periodic MACs makes some of the messages in the system not schedulable. We found that the bus utilization exceeds 100% with the addition of MACs for this benchmark. Also, the MAC and the periodic MAC approach with both $\rho_i = P_i$ and $\rho_i = 2P_i$ fail to converge while attempting to solve the recurrence relation and therefore do not find a response time for several of the messages. The RTA for the periodic MAC approach with $\rho_i = 10P_i$ converges but it still is not schedulable assuming implicit deadlines, which demonstrates the value of having the RTA formulation. We also considered an extreme case of the periodic MAC approach with $\rho_i = 1000P_i$; in this case, there are substantially diminishing returns from increasing the period of the MAC and the performance tracks quite closely with the case of $\rho_i = 10P_i$.

3) CAN Schedulability: For synthetic randomized schedulability experiments, we start the evaluation by creating 1000 message sets varying each message's period (implicit deadline) and the DLC. We assigned DLC (i.e., 1-8 bytes) and periods to each message uniformly at random from a period

[1]https://github.com/Embedded-Systems-Security-Lab/canasta.



(a) Schedulability



(b) Percent of messages meeting deadlines ($R_i < P_i$)

Fig. 6: Random message sets in Classical CAN with varying bus utilization and authentication schemes at 250 kbps bitrate

set containing 5, 10, 100, 1000, and 5000 ms, and varied the bus utilization from 10 to 90 percent. We generate the messages uniformly at random from the fixed period sets and the DLC range until reaching the utilization. Message IDs (priorities) are assigned rate monotonically. We calculate the CAN RTA for each generated message set as a baseline. We then calculate the MAC and periodic MAC approaches with the same message set. For the MAC approach, we add an authenticator using SecOC Profile 1 (24Bit-CMAC-8Bit-FV). If the resulting payload exceeds the 8 byte $D_{\max}$ of CAN then we generate an additional frame, i.e., $\beta_i + \alpha_i = 2$. For the periodic MAC approach we consider the addition of SecOC Profile 1 authenticators on every message at a periodic rate of $\rho_i = 2P_i$ and $\rho_i = 10P_i$. We also consider the case of $\rho_i = P_i$, which generates a separate authenticator frame for every message instance.

Fig. 6a shows the percentage of schedulable message sets for the different authentication schemes at 250 kbps CAN bitrate. Each datapoint shows the percent schedulable out of 1000 generated message sets for the given authentication approach at that bus utilization. As expected, the MAC and Periodic MAC with $\rho_i = P_i$ perform the worst; they both have a sharp drop-off around 30–40% bus utilization, which makes sense because they increase the number of message transmissions by 50–100%. At around 50% utilization the recurrence relation in the RTA for those approaches fails to converge. Somewhat surprisingly however the periodic approach with $\rho_i = 2P_i$ also tracks the same trend line, and even with $\rho_i = 10P_i$ the schedulability of the message sets drops at 40–50% utilization. We observe the same trend for the periodic MAC regardless of ρ_i because the ceiling function in

TABLE VI: BMW message set parameters, 100 kbps bitrate [29], with calculated worst-case response times for No MAC, MAC (Profile 1), and periodic MAC ($\rho_i \in 1 * P_i, 2 * P_i, 10 * P_i$).

ID (hex)	D_i	P_i (ms)	Description	C_{D_i} (ms)	No MAC	MAC	$\rho_i = P_i$	$\rho_i = 2 * P_i$	$\rho_i = 10 * P_i$
0x0A8	8	10	Torque, Clutch and Brake Status	1.35	2.7	3.65	4.05	4.05	4.05
0x0AA	8	10	Engine RPM and Throttle Position	1.35	4.05	5.95	6.35	6.35	6.35
0x0C0	2	200	ABS / Brake Counter	0.75	4.8	7.1	7.65	7.65	7.65
0x0CE	8	10	Individual Wheel Speeds	1.35	6.15	9.4	10.35	10.35	10.35
0x0D7	2	200	Counter (Airbag / Seatbelt related)	0.75	6.9	10.55	18.55	15.7	15.7
0x130	5	100	Ignition and Key Status (Terminal 15)	1.05	7.95	19.45	20.65	17.8	17.8
0x19E	8	200	ABS / Braking Force	1.35	9.3	28.65	30.15	20.4	20.4
0x1A6	8	100	Speed	1.35	10.65	37.85	39.35	29.6	26.75
0x1D0	8	200	Engine Temperature, Pressure Sensor and Handbrake	1.35	16.05	47.05	48.55	35.95	29.05
0x21A	3	5000	Lighting Status	0.85	16.9	48.3	49.95	37.35	30.45
0x26E	8	200	Ignition Status	1.35	18.25	57.5	59.55	40.05	37.2
0x335	8	1000	Unknown	1.35	19.6	59.8	68.75	49.25	39.5
0x349	5	200	Fuel Level Sensors	1.05	20.65	68.7	70.45	50.95	45.25
0x34F	2	1000	Handbrake Status	0.75	25.45	69.85	78.95	56.6	46.85
0x380	7	Once	VIN Number	1.25	26.7	78.95	88.35	56.6	49.35
0x39E	8	Once	Set Time and Date	1.35	28.05	88.15	90.75	68.4	55.8
0x3B4	8	4000	Battery Voltage and Charge Status	1.35	29.4	97.35	99.95	70.7	58.1
0x581	8	5000	Seat belt Status	1.35	29.4	98.3	100.9	75.7	59.05

TABLE VII: Modified SAE Benchmark [3] with calculated worst-case response times for No MAC, MAC (Profile 1), and periodic MAC ($\rho_i \in 1 * P_i, 2 * P_i, 10 * P_i$).

Sender	ID (hex)	D_i	P_i (ms)	C_{D_i} (ms)	No MAC	MAC	$\rho_i = P_i$	$\rho_i = 2 * P_i$	$\rho_i = 10 * P_i$
VC	A0	1	5	0.52	1.44	2.04	2.2	2.2	2.2
	B0	6	10	0.92	4.44	∞	∞	∞	15.6
	D0	1	1000	0.52	19.32	∞	∞	∞	89.44
Brakes	A1	2	5	0.6	2.04	4.12	3.56	3.56	3.56
	C1	1	100	0.52	9.44	∞	∞	∞	49.68
Battery	B2	1	10	0.52	4.96	∞	∞	∞	20.52
	C2	4	100	0.76	10.12	∞	∞	∞	79.48
	D2	3	1000	0.68	19.84	∞	∞	∞	99.8
Driver	A3	1	5	0.52	2.56	12.12	4.84	4.84	4.84
	B3	2	10	0.6	5.56	∞	∞	∞	29.52
IMC	A4	2	5	0.6	3.16	∞	∞	8.44	8.44
	B4	2	10	0.6	8.92	∞	∞	∞	45.64
Trans	A5	1	5	0.52	3.68	∞	∞	14.56	13.0
	C5	1	100	0.52	18.8	∞	∞	∞	88.92
	D5	1	1000	0.52	19.84	∞	∞	∞	100.32

Eq. 35, 38, and 41 can lead to the same output due to rounding up the division to the same value. Hence, the same number of instances or worst-case delay of a message can be observed across different settings of ρ_i.

To better understand the message set schedulability results, we also investigated the ratio of schedulable messages within the generated sets. As demonstrated in the SAE benchmark case study, often the messages with the shortest and longest periods can meet their deadlines despite authentication overhead, while those in the middle tend to fail first. Fig. 6b shows the percent of individual messages that meet their deadlines for the same generated message sets as the schedulability results from Fig. 6a. This plot shows more gradual performance degradation because many messages can still meet their deadlines. The periodic MAC approaches that skip more messages between authentications performs well until after 70% utilization, with $\rho_i = 2P_i$ failing to converge at 80% and $\rho_i = 10P_i$ at 90% utilization. These results indicate that a more intelligent approach to allocating MACs to messages could yield better real-time schedulability performance.

B. CAN FD Schedulability

We examine the performance impact of adding MACs to CAN FD using synthetic message sets to evaluate the RTA formulation. Similarly to CAN, we created 1000 message sets for CAN FD using DLC values ranging from 1 to 64 bytes and assigned periods from 5, 10, 100, 1000, and 5000 ms. We used a bus speed of 250kbps and 4Mbps for bitrate switching and calculated τ_{bit} and τ_{dbit} using Eq. (17) to generate their transmission times.

Fig. 7 shows the percentage of message sets that are schedulable, and we observe a drop-off in schedulability around 65% bus utilization for periodic MAC and $\rho_i = P_i$. The MAC and $\rho_i = 2P_i$ fail to converge at 90% bus utilization. We observe that the addition of MAC shows a moderate increase in the transmission times, with the 64-byte payload size incurring the most overhead since an extra frame is required to transmit the

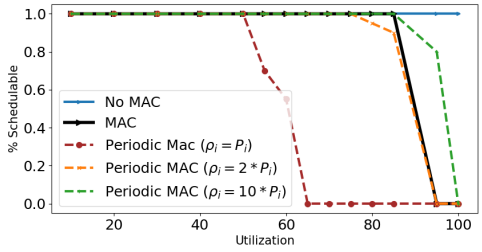


Fig. 7: Schedulability of random message sets for CAN FD with varying bus utilization and authentication schemes at $\tau_{bit} = 0.002$, $\tau_{dbit} = 0.00025$.

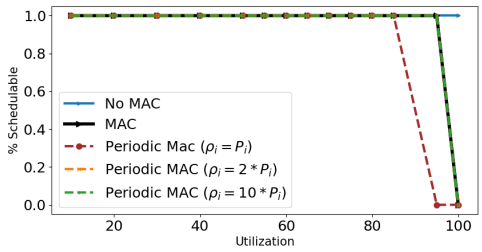


Fig. 8: Schedulability of random message sets for CAN XL with varying bus utilization and authentication schemes at $\tau_{bit} = 0.002$, $\tau_{dbit} = 0.0001$.

authenticator due to an already full data payload. Because the periodic MAC introduces an extra frame, the periodic MAC is not in general a good choice, however it does perform the best with sufficiently high enough period (e.g., $\rho_i = 10 * P_i$) at high bus utilization.

C. CAN XL Schedulability

For CAN XL, we generated 1000 message sets with DLC between 1 and 2048 and with the period set used for CAN and CAN FD. We use a bus speed of 500kbps and 10Mbps for bitrate switching. Fig. 8 shows the schedulability of randomized message sets, which demonstrates that adding MACs tracks closely with not using MACs until roughly 90% bus utilization. On one hand, the addition of MACs adds little overhead as they fit within the frame and the faster bitrate accommodates them. On the other hand, adding MACs at high bus utilization can cause total bus utilization to exceed 100% and therefore lose schedulability. For CAN XL, neglecting the computational overheads for MAC validation as we have done throughout, it seems that there is no advantage or disadvantage with respect to schedulability to use periodic MACs (with $\rho_i \geq 2 * P_i$) versus appending a MAC to every message.

IX. Conclusion

In this paper we provide the first comprehensive approach toward formalizing the impact that authentication schemes have on the real-time performance of messages over CAN, CAN FD, and CAN XL based on response time analysis. We show the utility of our approach by evaluating the effectiveness with respect to schedulability of periodic message authentication compared to authenticating every message instance across randomized message sets and for two CAN use cases. In this work, we have ignored the issues of key exchange and re-synchronization that are required by some authentication schemes [31]. Future work can tighten the bounds of our RTA in places where we have been overly pessimistic, and the RTA formulations can be extended to support evaluation of other proposed schemes for CAN authentication such as when transmitting the authenticator with separate IDs or using re-synchronization.

Acknowledgments

This work is partially supported by NSF CNS-2046705, NSF OAC-2001789, and Colorado Bill SB18-086.

References

[1] 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.

[2] K. Tindell, A. Burns, and A. J. Wellings, "Calculating controller area network (can) message response times," *Control engineering practice*, vol. 3, no. 8, pp. 1163–1169, 1995.

[3] K. Tindell, H. Hanssmon, and A. J. Wellings, "Analysing real-time communications: Controller area network (CAN)." in *RTSS*, 1994.

[4] C. AUTOSAR, "Specification of secure onboard communication," *AUTOSAR CP Release*, vol. 4, no. 1, 2017.

[5] G. Bloom, "WeepingCAN: A Stealthy CAN Bus-off Attack," in *Workshop on Automotive and Autonomous Vehicle Security*. Internet Society, Feb. 2021.

[6] B. Groza and P.-S. Murvay, "Security solutions for the controller area network: Bringing authentication to in-vehicle networks," *IEEE Vehicular Technology Magazine*, vol. 13, no. 1, pp. 40–47, 2018.

[7] B. Groza, S. Murvay, A. v. Herrewege, and I. Verbauwhede, "Libracan: A lightweight broadcast authentication protocol for controller area networks," in *International Conference on Cryptology and Network Security*. Springer, 2012, pp. 185–200.

[8] R. Kurachi, Y. Matsubara, H. Takada, N. Adachi, Y. Miyashita, and S. Horihata, "Cacan-centralized authentication system in can (controller area network)," in *14th Int. Conf. on Embedded Security in Cars (ESCAR 2014)*, 2014.

[9] M. Zhang, P. Parsch, H. Hoffmann, and A. Masrur, "Analyzing can's timing under periodically authenticated encryption," in *2022 Design, Automation & Test in Europe Conference & Exhibition (DATE)*, 2022, pp. 620–623.

[10] J. Daily, D. Nnaji, and B. Ettlinger, "Securing CAN Traffic on J1939 Networks," in *Workshop on Automotive and Autonomous Vehicle Security*. Internet Society, Feb. 2021.

[11] C.-W. Lin, Q. Zhu, C. Phung, and A. Sangiovanni-Vincentelli, "Security-aware mapping for can-based real-time distributed automotive systems," in *2013 IEEE/ACM International Conference on Computer-Aided Design (ICCAD)*. IEEE, 2013, pp. 115–121.

[12] B. Groza, L. Popa, and P.-S. Murvay, "Highly efficient authentication for can by identifier reallocation with ordered cmacs," *IEEE Transactions on Vehicular Technology*, vol. 69, no. 6, pp. 6129–6140, 2020.

[13] S. Nürnberger and C. Rossow, "–vatican–vetted, authenticated can bus," in *International Conference on Cryptographic Hardware and Embedded Systems*. Springer, 2016, pp. 106–124.

[14] J. Van Bulck, J. T. Mühlberg, and F. Piessens, "Vulcan: Efficient component authentication and software isolation for automotive control networks," in *Proceedings of the 33rd Annual Computer Security Applications Conference*, 2017, pp. 225–237.

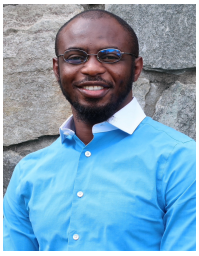
[15] G. Xie, L. T. Yang, W. Wu, K. Zeng, X. Xiao, and R. Li, "Security enhancement for real-time parallel in-vehicle applications by CAN fd message authentication," *IEEE Transactions on Intelligent Transportation Systems*, 2020.

[16] J. Petit and Z. Mammeri, "Impact of message authentication on braking distance in vehicular networks," in *Proc. of 5th ERST2 Workshop*, 2010.

[17] Z. Wu, J. Zhao, Y. Zhu, and Q. Li, "Research on vehicle cybersecurity based on dedicated security hardware and ecdh algorithm," SAE Technical Paper, Tech. Rep., 2017.

[18] Y. Xie, Y. Guo, S. Yang, J. Zhou, and X. Chen, "Security-related hardware cost optimization for CAN fd-based automotive cyber-physical systems," *Sensors*, vol. 21, no. 20, p. 6807, 2021.
[19] V. Lesi, I. Jovanov, and M. Pajic, "Security-aware scheduling of embedded control tasks," *ACM Transactions on Embedded Computing Systems (TECS)*, vol. 16, no. 5s, pp. 1–21, 2017.
[20] S. K. Ghosh, J. S. RC, V. Jain, and S. Dey, "Reliable and secure design-space-exploration for cyber-physical systems," *ACM Transactions on Embedded Computing Systems (TECS)*, vol. 19, no. 3, pp. 1–29, 2020.
[21] U. D. Bordoloi and S. Samii, "The frame packing problem for can-fd," in *2014 IEEE Real-Time Systems Symposium*. IEEE, 2014, pp. 284–293.
[22] R. De Andrade, K. N. Hodel, J. F. Justo, A. M. Laganá, M. M. Santos, and Z. Gu, "Analytical and experimental performance evaluations of can-fd bus," *IEEE Access*, vol. 6, pp. 21 287–21 295, 2018.
[23] T. Nolte, H. Hansson, and C. Norström, "Minimizing CAN response-time jitter by message manipulation," in *Proc. IEEE Real-Time and Embedded Technology and Applications Symposium*, 2002, pp. 197–206.
[24] G. Cena, I. Cibrario Bertolotti, T. Hu, and A. Valenzano, "Performance comparison of mechanisms to reduce bit stuffing jitters in Controller Area Networks," in *Proc. 17th IEEE Conference on Emerging Technologies and Factory Automation (ETFA)*, Sep. 2012, pp. 1–8.
[25] I. Broster, A. Burns, and G. Rodriguez-Navas, "Timing analysis of real-time communication under electromagnetic interference," *Real-Time Systems*, vol. 30, no. 1-2, pp. 55–81, 2005.
[26] ISO, *ISO 11898-1 – Road vehicles – Controller area network (CAN) – Part 1: Data link layer and physical signalling*, 2nd ed., International Organization for Standardization, Dec. 2015.
[27] CiA, *Draft specification CiA 610-1 version 1.0.0 – CAN XL specifications and test plans – Part 1: Data link layer and physical coding sub-layer requirements*, CAN in Automation (CiA) e. V., Mar. 2022.
[28] M. Dworkin, "Recommendation for Block Cipher Modes of Operation: the CMAC Mode for Authentication," National Institute of Standards and Technology, Tech. Rep. NIST Special Publication (SP) 800-38B, Oct. 2016. [Online]. Available: https://csrc.nist.gov/publications/detail/sp/800-38b/final
[29] R. Buttigieg, M. Farrugia, and C. Meli, "Security issues in controller area networks in automobiles," in *2017 18th International Conference on Sciences and Techniques of Automatic Control and Computer Engineering (STA)*. IEEE, 2017, pp. 93–98.
[30] K. Tindell and A. Burns, "Guaranteeing message latencies on control area network (can)," in *Proceedings of the 1st International CAN Conference*, 1994.
[31] A.-I. Radu and F. D. Garcia, "Leia: A lightweight authentication protocol for can," in *European Symposium on Research in Computer Security*. Springer, 2016, pp. 283–300.

Omolade Ikumapayi is a Ph.D. student in the Computer Science Department at University of Colorado Colorado Springs. Her research focuses on the security of real-time systems.



Habeeb Olufowobi (M'20) received his Ph.D. in computer science from Howard University in 2019. He joined University of Texas at Arlington as Assistant Professor of Computer Science and Engineering in 2020. His research focuses on embedded systems security and privacy challenges in emerging network technologies for connected autonomous vehicles, the Internet of Vehicles (IoV) in a smart city ecosystem, and vehicular cloud network.

Jeremy Daily is an Associate Professor of Systems Engineering at Colorado State University in Fort Collins, CO. He helps run the CyberTruck Challenge, CyberAuto Challenge, CyberBoat Challenge. He also provides instruction for the CyberTractor Challenge. Before coming to CSU, he was an Associate Professor of Mechanical Engineering at the University of Tulsa. Dr. Daily's research interests include heavy vehicle cybersecurity and digital forensics for transportation systems.

Tingting Hu received her Master degree in Computer Engineering in 2010 and PhD degree in Computer and Control Engineering in 2015 from Politecnico di Torino, Turin, Italy. Between 2010 and 2016, she also worked as a research fellow with the National Research Council of Italy (CNR), Turin, Italy. She joined University of Luxembourg as a post-doc researcher in 2016 and research scientist in 2019 with the Faculty of Science, Technology and Medicine (FSTM). Her research interest mainly concerns real-time embedded systems with a focus on real-time networks and E/E architecture design for Software Defined Vehicles (SDV). She serves as program committee member and technical referee for several primary international conferences and journals.

Ivan Cibrario Bertolotti (M'06) received the Laurea degree (summa cum laude) in computer science from the University of Torino, Turin, Italy, in 1996. Since then, he has been a Researcher with the National Research Council of Italy (CNR), Rome, Italy. Currently, he is with the Institute of Electronics, Computer and Telecommunication Engineering (IEIIT), Turin. He has written text books on real-time embedded systems and taught several courses on real-time operating systems at Politecnico di Torino, Turin, and he serves as a Technical Referee for primary international journals and conferences. His research interests include real-time operating system design and implementation, industrial communication systems and protocols, as well as modeling languages and runtime support for cyber-physical systems.

Gedare Bloom (SM'19) received his Ph.D. in computer science from The George Washington University in 2013. He joined the University of Colorado Colorado Springs as Assistant Professor of Computer Science in 2019 and Associate Professor in 2022. He was Assistant Professor of Computer Science at Howard University from 2015-2019. His research expertise is computer system security with emphasis on real-time embedded systems. He is an associate editor for the IEEE TRANSACTIONS ON VEHICULAR TECHNOLOGY.