Design and Performance Analysis of Secure and Dependable Cybercars: A Steer-by-Wire Case Study

Arslan Munir
Computer Science & Engineering Department
University of Nevada, Reno, NV, USA
Email: arslan@unr.edu

Farinaz Koushanfar
Electrical & Computer Engineering Department
Rice University, Houston, TX, USA
Email: farinaz@rice.edu

***Abstract*—The next generation of automobiles (also known as cybercars) will increasingly incorporate electronic control units (ECUs) in novel automotive control applications. Recent work has demonstrated vulnerability of modern car control systems to security attacks that directly impact the cybercar's physical safety and dependability. In this paper, we provide an integrated approach for the design of secure and dependable cybercars using a case study: a steer-by-wire (SBW) application over controller area network (CAN). The challenge is to embed both security and dependability over CAN while ensuring that the real-time constraints of the cybercar applications are not violated. Our approach enables early design feasibility analysis by embedding essential security primitives (i.e., confidentiality, integrity, and authentication) over CAN subject to the real-time constraints imposed by the desired quality of service and behavioral reliability. Our method leverages multi-core ECUs for providing fault-tolerance by redundant multi-threading (RMT) and also further enhances RMT for quick error detection. We quantify the error resilience of our approach and evaluate the interplay of performance, fault-tolerance, security, and scalability for our SBW case study.**

***Keywords*—Automotive embedded systems, multi-core, security, dependability, fault-tolerance, x-by-wire, steer-by-wire**

I. Introduction and Motivation

Modern cars consist of more than 100 electronic control units (ECUs) to implement various distributed control applications. The next generation of automobiles (also known as cybercars) will further escalate the proliferation of ECUs to enable new and exciting control and infotainment applications. The most prevalent protocol for communication among the ECUs in these distributed control applications is controller area network (CAN), which has already been implemented in billions of devices and long-lived transportation systems, and thus likely to stay for many years to come. Emerging automotive distributed control applications include *x-by-wire*, e.g., break-by-wire, steer-by-wire (SBW), where the electronic controllers substitute the traditional mechanical and/or hydraulic systems.

The realization of emerging cybercar applications require addressing dependability and security issues. The cybercar applications have stringent dependability requirements as stipulated by ISO 26262 [1], which require tolerance of at least one critical fault without loss of functionality [2]. Meeting these cybercar dependability requirements poses various challenges. Harsh operating environments coupled with external noise and radiation render automotive electronic systems vulnerable to permanent and transient faults. While permanent faults can impair or stop the correct functionality of the system, *soft errors* induced by transient faults can remarkably reduce system availability. Furthermore, automotive electronic components are susceptible to security vulnerabilities. Since CAN messages are transmitted in plain-text format, intruders may be able to gain access and even alter the CAN messages creating security threats. These security threats are exacerbated by the increasing integration of cybercar applications with external entities such as consumer electronics, other vehicles, and networks.

The cyber-physical attributes of modern automotive systems directly couple security vulnerabilities to the cybercar's physical safety and dependability: an attacker who is able to infiltrate any ECU can potentially circumvent many safety-critical systems while completely ignoring the driver input [3]. Simultaneous integration of security and dependability in cybercars is challenging. The biggest challenge in the simultaneous integration of security and safety is to avoid violation of the cybercar application's hard real-time constraints. Timeliness (meeting timing constraints) is perceived as the system's quality of service (QoS), commonly considered as a performance measure. We emphasize that the system's QoS must be considered as a dependability measure that can impact the system's availability and safety, beyond a certain *critical threshold* as the driver can totally lose the control of his/her car beyond that critical threshold [4].

The security vulnerabilities and dependability requirements of cybercars warrant inclusion and performance analysis of security and dependability approaches for safety-critical applications. Although earlier work in automotive systems has addressed certain aspects of security and dependability (e.g., [4], [5]), the interplay between performance, security, and safety has not yet been explored. To overcome the limitations of earlier work, we provide a new comprehensive approach to the design and analysis of secure and dependable cybercars with SBW application as a case study. Our main technical contributions are as follows:

- A novel integrated approach that enables early design phase feasibility and performance analysis for devising secure and dependable cybercars. The approach is demonstrated on a SBW application case study.
- Embedding and analyzing security primitives over CAN while adhering to the stringent real-time constraints for safety-critical cybercar applications. In our analysis, we exploit advanced encryption standard (AES) for providing message confidentiality

and hash-based message authentication code (HMAC) for ensuring message authenticity and integrity.

- Proposal to include multi-core ECUs (instead of conventional single-core ones) to meet the fault-tolerance (FT) requirements (one permanent and multiple soft-errors) under stringent cost and real-time constraints by redundant multi-threading (RMT). We denote this FT approach by *FT-RMT*.
- Further enhancement of the RMT by exploiting quick error detection (QED) [6]. We denote this enhanced FT approach by *FT-RMT-QED*.
- Quantification of the error resilience of our approach by calculating the number of tolerable computational and transmission errors that are manifested as response time variations under the stringent QoS constraints.
- Analyzing the scaling properties of our secure and dependable SBW system under different CAN bus load conditions and message priority assignments.

The scope of this work is to ensure that simultaneous integration of security and dependability primitives in cybercars does not violate stringent real-time constraints imposed by the desired QoS. Although our proposed approach leverages existing security and dependability techniques, cardinal novelty of our work lies in the simultaneously integration of both dependability and security, and the performance and feasibility analysis of our integrated approach over CAN. The evaluation metric for our feasibility analysis is response time, which is constrained by the desired QoS.

II. Related Work

Security for automotive embedded systems has been studied in literature. Checkoway et al. [7] analyzed the external attack surface of a modern automobile. The authors discovered that remote exploitation is possible via a broad range of attack vectors such as mechanics tools, CD players, bluetooth, and cellular radio. Chávez et al. [5] incorporated confidentiality service in CAN based on RC4. However, the authors did not consider FT aspects or CAN message priorities while analyzing the security over CAN.

Several earlier works explored dependability for automotive embedded systems. Baleani et al. [8] discussed various FT architectures for automotive applications including lock-step dual processor architecture, loosely-synchronized dual processor architecture, and triple modular redundant architecture. Although, the authors compared various FT architectures' costs based on the area estimates, the study did not quantify the architectures' FT capabilities subject to real-time constraints. Salewski et al. [9] compared fault-handling measures for microcontrollers and field-programmable gate arrays (FPGAs). However, the work did not study FT for distributed applications where both computation and communication errors affect dependability.

Zeng et al. [10] computed the probability distribution of CAN message response times using statistical analysis when only partial information about the functionality and architecture of a vehicle was available. Although the statistical model can predict CAN message response times, however, many parameters required by the model, such as queueing lengths and the time between consecutive higher priority message bursts, are difficult to determine. Additionally, the limitation of statistical methods for high-priority messages motivated us for rigorous simulations of SBW systems to provide better response time estimates than the statistical methods.

Previous work investigated automotive x-by-wire systems for future cybercars. Schweppe et al. [11] studied an active braking system assisted by vehicle-2-X communication. Wilwert et al. [4] presented an approach for evaluating the temporal performance and behavioral reliability of an SBW system considering the delay variation introduced by network transmission errors. The authors evaluated the maximum tolerable system response time and the impact of this response time on QoS for a time division multiple access communication protocol. However, the work considered only communication transmission errors and not the computational errors in ECUs. Furthermore, the authors did not consider the system vulnerability to security attacks. In our work, we consider security overhead while analyzing the system response time. Moreover, the previous work did not analyze the scalability aspects whereas our work analyzes the SBW system's response time under different load conditions and message priority assignments.

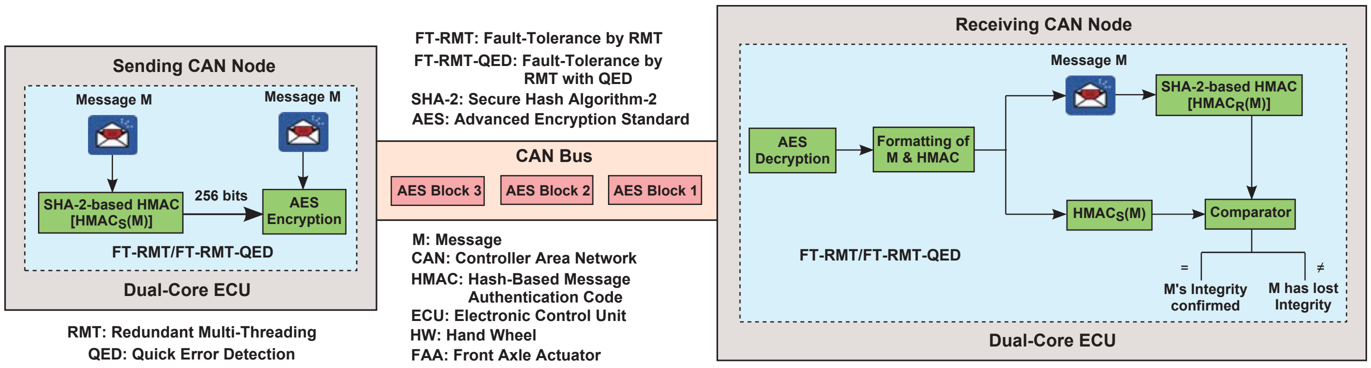
III. A Dependable and Secure Approach for Cybercar Design

The design of secure and dependable cybercars is challenging because of limited resource budgets (e.g., memory and processing, bandwidth, cost) and real-time constraints. In particular, the inclusion of dependability and security primitives, and protocols must not violate the real-time constraints. The scope of our approach is confined to error detection and correction for dependability, and to provide confidentiality, integrity, and authentication for security. Fig. 1 provides an overview of our dependable and secure approach for cybercar design. The figure shows the operations involved at both the sending and receiving CAN nodes to ensure dependability and security. This section elaborates the dependability and security primitives adopted in our approach.

A. Dependability

To assist the design and production of safe automotive systems, International Organization for Standardization (ISO) has developed a functional safety standard, viz., ISO 26262 [1]. Current automotive systems consist of single-core ECUs that are unable to meet the FT requirements of cybercars as outlined in ISO 26262. To exploit the technological advancements in silicon and accompanied low-cost of single-chip solutions, our approach leverages multi-core ECUs to provide FT. Our approach considers a generic dual-core architecture that meets the low cost as well as flexibility requirements since the architecture can adapt to various performance and FT requirements for an application. We point out that the dependability approaches based on specific hardware architectures such as lock-step dual processor architecture are expensive as compared to the generic dual-core architectures and offer limited flexibility.

We explore two FT approaches leveraging RMT on the dual-core architecture. The first one is FT-RMT, which executes safety-critical computations on redundant threads and



Sending CAN Node
Message M
Message M
SHA-2-based HMAC [HMACS(M)]
256 bits
AES Encryption
FT-RMT/FT-RMT-QED
Dual-Core ECU
RMT: Redundant Multi-Threading
QED: Quick Error Detection
FT-RMT: Fault-Tolerance by RMT
FT-RMT-QED: Fault-Tolerance by RMT with QED
SHA-2: Secure Hash Algorithm-2
AES: Advanced Encryption Standard
CAN Bus
AES Block 3
AES Block 2
AES Block 1
M: Message
CAN: Controller Area Network
HMAC: Hash-Based Message Authentication Code
ECU: Electronic Control Unit
HW: Hand Wheel
FAA: Front Axle Actuator
Receiving CAN Node
Message M
SHA-2-based HMAC [HMACR(M)]
AES Decryption
Formatting of M & HMAC
HMACS(M)
Comparator
FT-RMT/FT-RMT-QED
=
≠
M's Integrity confirmed
M has lost Integrity
Dual-Core ECU

Fig. 1: A dependable and secure approach for cybercar design.

detects an error at the end of the computation if there is a mismatch between the two threads' output. The second approach is FT-RMT-QED that enhances FT-RMT with QED [6]. In the FT-RMT-QED, the main thread executes original instructions and check instructions, which are inserted at different points in the program/computation, whereas another thread executes duplicated instructions. When an error is detected earlier by FT-RMT-QED rather than at the end of the computation, the erroneous computation is aborted, and the computation is restarted to obtain an error-free result. Error detection latencies in the FT-RMT-QED approach are configurable and can range from a few cycles to a few thousand cycles depending on the desired tradeoff between error detection latency and complexity (i.e., additional software modifications to incorporate QED checks).

Our FT approaches on dual-core ECUs can tolerate one permanent fault and multiple soft-errors. Multiple soft-errors are tolerated as our approach recomputes the result on a soft-error detection at any point in the program, and repeats this re-computation process until an error-free result is obtained. A major advantage of both the FT-RMT and the FT-RMT-QED approaches is that the methods are *self-checking*, i.e., they do not require a separate golden response created through simulation.

B. Security Threat Model

In order to better elucidate our security approach, we characterize the likely capabilities of an adversary aiming to infiltrate automobile's internal networks (e.g., CAN, FlexRay). Modern automobiles provide several interfaces, such as on-board diagnostics port (OBD-II), entertainment systems (e.g., CD, USB, iPod), and short range or long range wireless access, that provide direct or indirect access to automobile's internal networks [7]. Assuming that the adversary has gained access to the car's internal networks either directly or indirectly, this section summarizes briefly the associated security threat model against which our approach provides resilience.

Threat 1—Passive Eavesdropping & Traffic Analysis: An adversary can sniff and store all the traffic in a car's internal network.

Threat 2—Active Eavesdropping & Message Injection: An adversary can generate any chosen message as well as modify the contents of a message.

We reemphasize that current automotive protocols do not incorporate any security primitives to countermeasure the above mentioned threats. *Threats 1* and *2* are possible in the absence or possible breaking of data confidentiality and integrity over automotive internal networks. These threats expose the automobile to severe vulnerabilities as the adversary can potentially circumvent many safety-critical systems (e.g., brakes, engine, lights, locks) while completely ignoring the driver's input [3].

C. Security

To countermeasure the threats presented in Section III-B, our approach provides confidentiality, integrity, and authentication for CAN messages. To minimize the encryption overhead while providing adequate security for CAN message lifetimes, we leverage AES-128 (128-bit) encryption to provide confidentiality and an HMAC based on SHA-2/SHA-256 (Secure Hash Algorithm-2) to render integrity and authenticity. The SHA-2-based HMAC module implements SHA-256 algorithm that takes the message M as input at the sending CAN node and outputs 256 bits, which is known as the *message digest*. The 256-bit HMAC $HMAC_S(M)$ and the message M are given as input to the AES encryption module, which encrypts the message and $HMAC_S(M)$. Our approach assumes that initial AES and HMAC keys are stored in secure tamper resistant memories of participating ECUs by original equipment manufacturers (OEMs). Furthermore, the AES and HMAC keys are updated/refereshed deterministically over time by participating ECUs as is done in transport layer security (TLS) [12]. Our approach for providing confidentiality, integrity, and authentication is inspired by secure sockets layer (SSL) [12]. Hence, confidentiality, integrity, and authentication can be added to a message M as

$$\begin{aligned}\mathcal{O}_{M,S} &= \text{calc}\,[HMAC_S(M)] \\ &\quad + \text{Encrypt}_{AES}\,[M + HMAC_S(M)] \qquad (1)\end{aligned}$$

where function `calc()` denotes calculation of $HMAC_S(M)$ of the message M by the sending node S, and $\mathcal{O}_{M,S}$ denotes operations at the sending CAN node S that include encryption of the message M and the $HMAC_S(M)$ using AES algorithm.

To determine the storage bits required for the $\mathcal{O}_{M,S}$ operation, let us consider an 8-byte CAN message. Hence, $M + HMAC(M)$ = 64 + 256 bits = 320 bits. The encryption of these 320 bits require three 128-bit AES blocks. The first two AES blocks encrypt the first 256 bits whereas the third AES block encrypts the remaining 64 bits padded by a 1 bit followed by 0 bits to make the block length of 128 bits (padding in hexadecimal looks like 0x80,0x00,. . .,0x00). The transfer of 384 bits (3 AES blocks) of the encrypted message requires 384/64 = 6 CAN message frames. Hence, an unencrypted 8-byte CAN message requires six 8-byte CAN messages on encryption. We propose to use packet sequence number (ranging from zero to five) of an encrypted message in the three most significant bits of the data field of each CAN frame. This packet sequence number would help in the message assembling and for retransmission request of an erroneous frame in case of transmission errors.

Eq. (1) summarizes the operations at the sending CAN node, however, additional comparison instructions are required to implement our FT approaches. For instance, the FT-RMT-QED requires comparison instructions to compare the threads' output for error detection. The operations at the sending CAN node for the FT-RMT-QED can be given as

$$\begin{aligned}\mathcal{O}_{M,S}^{FT-RMT-QED} &= \text{calc}^{\forall i=1,2}\left[HMAC_S^{T_i}(M)\right] \\ &+ \text{Encrypt}_{\text{AES block}_j}^{\forall j=1,2,3}\left[M + HMAC_S^{T_i\ \forall\ i=1,2}(M)\right] \quad (2)\end{aligned}$$

where $HMAC_S^{T_i}(M)$ denotes the HMAC calculated by thread $T_i \ \forall\ i = 1, 2$ at the sending CAN node.

Fig. 1 shows that the receiving CAN node consists of the following modules: AES decryption module, formatting module, SHA-2-based HMAC module, and a comparator module. The AES decryption module decrypts the received CAN frames. The formatting module operates on the decrypted CAN frames to retrieve the message M and $HMAC_S(M)$. SHA-2-based HMAC module calculates the HMAC of the received message $HMAC_R(M)$. To verify the integrity and authenticity of the received message, the comparator module at the receiving CAN node compares $HMAC_S(M)$ with $HMAC_R(M)$. If $HMAC_S(M)$ is equal to $HMAC_R(M)$, then the received message is authentic otherwise the message has lost its integrity. The operations at the receiving CAN node can be summarized as

$$\begin{aligned}\mathcal{O}_{M,R} &= \text{Decrypt}_{AES}\left[M + HMAC_S(M)\right] \\ &+ \text{format}\left[M + HMAC_S(M)\right] + HMAC_R(M) \\ &+ \text{comp}\left[HMAC_S, HMAC_R\right] \quad (3)\end{aligned}$$

where $\mathcal{O}_{M,R}$ denotes the message decryption along with the associated operations performed at the receiving CAN node to verify the received message's integrity. The function `format()` denotes the formatting/extraction of message M and $HMAC_S(M)$ from the received CAN frames, and the function `comp()` denotes the comparison of $HMAC_S(M)$ with $HMAC_R(M)$. Equations describing the operations at the receiving CAN node (similar to Eq. (2) for the sending CAN node) for FT-RMT and FT-RMT-QED can be written based on Eq. (3).

Our security approach provides resilience against the security threat model described in Section III-B. There is currently no known analytical attack against AES and a brute-force attack leveraging a supercomputer (10.51 petaFLOPS) would require 1 billion billion (10^{18}) years to crack the 128-bit AES key [13]. There are also currently no known collisions or attacks against SHA-2. With reference to *threat 1*, an adversary may eavesdrop on traffic but the adversary cannot decrypt the packets without knowledge of the key. Our approach completely eliminates *threat 2* because SHA-2-based HMAC not only prevents insertion of forged messages but also prohibits message modifications. Refreshing of AES and HMAC keys over time by ECUs in our approach further prevents replay attacks.

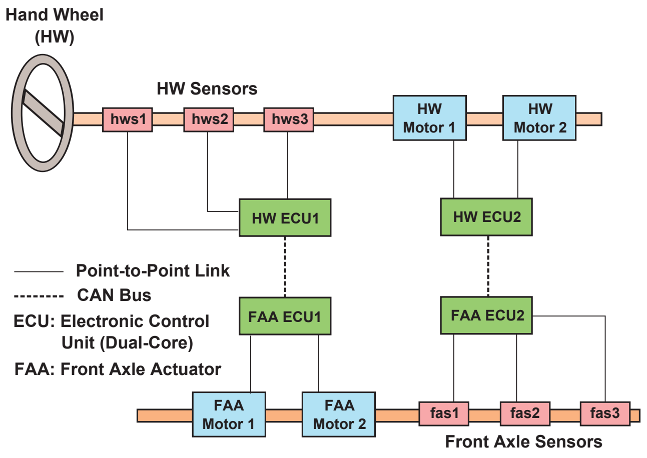


Fig. 2: Steer-by-wire operational architecture.

IV. Steer-by-Wire System

An SBW system provides various advantages over mechanical steering systems that motivate the adoption of SBW systems for cybercars. An SBW system eliminates the risk of steering column entering the cockpit in the event of a frontal crash and thus enhances safety. Since steering column is one of the heaviest components in the vehicle, removing the steering column reduces the weight of the vehicle and therefore lessens fuel consumption. SBW systems enhance driver comfort by providing a variable steering ratio. This section elaborates our SBW case study that leverages dual-core ECUs to incorporate dependability and security primitives.

A. Steer-by-wire operational architecture

Fig. 2 depicts the SBW architecture that we consider for our case study. The architecture provides FT by redundancy at ECU-level, sensor-level, and actuator level. The architecture consists of two dual-core hand wheel ECUs (HW ECU1 and HW ECU2) and two dual-core front axle actuator ECUs (FAA ECU1 and FAA ECU2). Each of the ECUs is connected to the CAN bus. Our SBW architecture consists of three hand wheel sensors (hws1, hws2, and hws3) that are placed near the hand wheel to measure the driver's requests in terms of hand wheel angle, hand wheel torque, and the hand wheel speed. Similarly, three front axle sensors (fas1, fas2, and fas3) measure the front axle position. Both the hand wheel sensors (the front axle sensors) are connected to the HW ECUs (FAA ECUs) by point-to-point links. Two front axle actuator (FAA) motors (FAA motor 1 and FAA motor 2) operate in active redundancy on the front axle while two hand wheel (HW) motors (HW motor 1 and HW motor 2) operate in active redundancy on the hand wheel.

An SBW system aims to provide two main services [2]: 1) Front axle control (FAC) that controls the wheel direction in accordance with the driver's request, and 2) Hand wheel force feedback (HWF) that provides a mechanical-like force feedback to the hand wheel.

1) Front axle control function: The front axle control (FAC) function computes the orders that are given to the front axle motor(s) according to the state of the front axle and the driver's requests obtained through the hand wheel. The three HW sensors (hws1, hws2, and hws3) measure the driver's requests and send these measurements to HW ECU1. The HW ECU1 cores operate either in the FT-RMT or the FT-RMT-QED mode to perform the necessary filtering and computations on the sensed input. After the necessary computations, HW ECU1 cores format the computed orders/signals in CAN message format, calculate message's HMAC to provide integrity, and perform encryption of the message and the associated HMAC to provide confidentiality. The encrypted messages are sent by the HW ECU1 CAN controllers on the CAN bus to the FAA ECU1. The FAA ECU1 placed behind the front axle consumes this data as well as the last wheel position to determine the commands for the FAA motor 1 and 2.

2) Hand wheel force feedback function: The hand wheel force feedback (HWF) function computes the orders that are given to the hand wheel motor(s) according to the speed of the vehicle, the front axle position, and the front tie rod force. The three front axle sensors (fas1, fas2, and fas3) send the sensed front axle position to the FAA ECU2. The FAA ECU2 cores perform the necessary computations, format the computed signals/values in CAN message format, calculate HMAC, and perform encryption of the message and the associated HMAC. The FAA ECU2 CAN controller sends the encrypted messages on the CAN bus, which are then consumed by the HW ECU2 to compute the commands for HW motor 1 and 2 to provide the necessary force feedback to the HW.

B. QoS and Behavioral Reliability

An SBW system is sensitive to the delay from the driver's request at the hand wheel to the reception of the request by the front axle actuators. This end-to-end delay/response time T_{res} is perceived as the QoS, and can impact availability and safety in the worst case if exceeded beyond a *critical threshold* T_{max}. The *behavioral reliability* is defined as the probability that the worst-case response time is less than the critical threshold [4]. Automotive OEMs determine T_{max} by Matlab/Simulink simulations and vehicle tests. The impact of T_{res} variation on vehicle's performance and stability can be evaluated in terms of a *QoS score*, denoted by S, determined by the time required to reach the desired position and stability. There exists a linear relation between S and the perturbation time (response time) for instantaneous rotation of handwheel. Research reveals that with a minimum tolerable S of 11.13, the critical limit T_{max} for the response time is 11.5 ms [4] beyond which the vehicle becomes unstable and the safety of the driver can be at risk.

In the following, we analytically model the response time for the SBW functions. We also model the error resilience provided by our adopted FT approaches (Section III) for the SBW system subject to the implicit timing constraints imposed by behavioral reliability.

The T_{res} for the FAC function (HWF function) comprises of the *pure delay* T_p, the *mechatronic delay* T_{mech}, and the *sensing delay* T_{sens}, i.e.,

$$T_{res} = T_p + T_{mech} + T_{sens} \tag{4}$$

The pure delay comprises of ECUs' computational delay for processing the driver's command, producing the actuator's command, and the transmission delay including the bus arbitration. The pure delay for our SBW system also consists of security and dependability primitives processing. The mechatronic delay is the delay introduced by actuators such as electric motors. The sensing delay corresponds to the delay involved in sensing the driver's command (front axle position) and storing the sensed information in a memory location accessible by HW ECUs (FAA ECUs). Since T_{mech} and T_{sens} can be upper bounded by a constant (3.5 ms in most cases [14]), we focus on T_p for our analysis [2]. The critical limit for pure delay T^p_{max} is 8 ms corresponding to T_{max} of 11.5 ms [4]. Systems that cannot guarantee a T_p lower than a tolerable upper bound T^p_{max} are considered unstable. The behavioral reliability P_{BR} evaluation is based on the worst-case T_P and not its nominal value because of the safety-critical nature of the system, i.e., $P_{BR} = P[T_{P_{WC}} < T^p_{max}]$, where $T_{P_{WC}}$ denotes the worst-case T_P. The nominal T_P for the FAC function, T_P^{FAC}, is given as

$$T_P^{FAC} = T_{ecu-hw1} + T_{ecu-faa1} + T^{hw1-faa1}_{channel}, \tag{5}$$

where $T_{ecu-hw1}$ and $T_{ecu-faa1}$ denote the computation time at HW ECU1 and FAA ECU1, respectively, and $T^{hw1-faa1}_{channel}$ denotes the channel time to transmit secure messages from HW ECU1 to FAA ECU1. The worst-case T_P for the FAC function, T_P^{FAC}, is given as

$$\begin{aligned} T^{FAC}_{P_{WC}} = n1 \cdot T_{ecu-hw1} + n2 \cdot T_{ecu-faa1} + n3 \cdot T^{hw1-faa1}_{channel}, \\ n1, n2 \in \{1, 2, \ldots\},\ n3 \in \{1, 1.167, 1.33, 1.5, \ldots\}, \end{aligned} \tag{6}$$

where $n1$ and $n2$ denote the number of computations required to yield an error-free result at HW ECU1 and FAA ECU1, respectively, and $n3$ denotes the number of transmissions required for error-free sending of secure messages over CAN. $n3$ values follow a fractional variation as six encrypted CAN frames are required for one unencrypted message in our SBW system and errors can occur in transmission of any or all frames.

For ensuring the QoS dictated by behavioral reliability, the $T_{P_{WC}}$ must be less than or equal to T^p_{max}. Hence, for the FAC function

$$n1 \cdot T_{ecu-hw1} + n2 \cdot T_{ecu-faa1} + n3 \cdot T^{hw1-faa1}_{channel} \leq T^p_{max} \tag{7}$$

Eq. (7) helps in analyzing the number of computational and transmission errors tolerated by a secure and dependable SBW system to attain the desired QoS and behavioral reliability corresponding to T^p_{max}.

Equation (7) indicates that early detection of errors in the program can provide room for more computations to yield the error-free result within the time constraint dictated by the required QoS. For FT-RMT-QED, Equation (7) gives an estimate for the maximum number of tolerable errors and the exact number of maximum tolerable errors is determined by inserting the computation time value for the error detected at a particular point in the program. For example, using FT-RMT-QED and keeping $n2$ and $n3$ fixed, actual $n1 \cdot T_{ecu-hw1}$ is determined as

$$n1 \cdot T_{ecu-hw1} = \begin{cases} T^{C}_{ecu-hw1}, & n1 = 1 \\ (n1-1) \cdot T^{QED}_{ecu-hw1} + T^{C}_{ecu-hw1}, & n1 \geq 2. \end{cases} \quad (8)$$

where $T^{C}_{ecu-hw1}$ denotes the time required for the complete computation at HW ECU1 whereas $T^{QED}_{ecu-hw1}$ denotes the error detection time for an erroneous computation at HW ECU1. Equation (8) assumes that all errors are detected at the same point in the program using QED, however, if errors are detected at different points, error detection times for the errors detected at different points in the program are to be added accordingly. The nominal and worst-case pure delay, and the number of maximum tolerable errors for the HWF function can be derived similarly.

V. Evaluation Results

To experimentally evaluate our proposed approach, we implement security and dependability primitives on an Intel quad-core processor, with symmetric multiprocessor architecture, and obtain the clock cycles required for execution where dependability is enabled by FT-RMT and FT-RMT-QED on dual cores. The Intel processor runs `GNU/Linux 2.6.18-308.24.1.el5PAE` version #1 `SMP`. Using the clock cycles, we estimate the security primitives execution time on a 32-bit multi-core ECU for safety-critical cybercar applications. We adopt OpenMP [15] to provide RMT-based FT on a multi-core architecture. We are further verifying our security and dependability primitives implementation on Freescale's MPC5777M ECU (quad-core 32-bit ECU) [16]. For our case study, we assume the steering wheel sensor sampling rate to be fixed at 420 Hz, i.e., T_{sens} = 2.38 ms [14]. We simulate our SBW system in Vector CANoe [17] with CAN baud rate set to 1 Mbps. We use CAPL (Vector CANoe programming language) to implement the SBW functions on ECUs. This section first presents the timing analysis for implementing security and dependability primitives on a 32-bit multi-core ECU. We then quantify the number of computational faults tolerated by ECUs for the SBW system with given QoS and behavioral reliability constraints. Finally, the section presents feasibility and scalability analysis for the SBW system as the CAN bus load varies.

A. Timing Analysis

For timing analysis of our FT approaches, we inject soft errors at different points in the program (security primitives implementation). Our software-based fault injection emulates bit flipping in the program/memory due to external noise and/or radiation. Table I presents the timing results with FT operational modes (FT-RMT and FT-RMT-QED) for a 32-bit ECU operating at 200 MHz for the $\mathcal{O}_{M,S}$ computations at the sending CAN node given by Eq. (1). We measure the clock cycles (averaged over 10 runs to smooth any discrepancies due to operating system overheads) using `rdtsc()` [18] at the start and end of computations. Table I indicates that incorporating FT (in any configuration: FT-RMT or FT-RMT-QED) incurs performance overhead as compared to the single-core implementation with no FT (NFT). For example, FT-RMT and FT-RMT-QED incur performance overheads of 36% and 41%, respectively, at the sending CAN node. The FT techniques incur performance penalty due to inherent multi-threading overhead, and additional instructions for comparison operations. Results verify that the FT-RMT-QED enables earlier detection (and/or correction) of errors for $\mathcal{O}_{M,S}$ computations as compared to the FT-RMT depending on the error point in the program. For example, the FT-RMT-QED detects an error 64,805 clock cycles earlier (an improvement of 1.4x) as compared to the FT-RMT when the error occurs at HMAC calculation.

TABLE I: $\mathcal{O}_{M,S}$ timing results for a 32-bit ECU @ 200 MHz.

Operational Mode	Error Detection Point	Clock Cycles	Time (μs)
NFT	N/A	163,218	816.09
FT-RMT	@ end of computation	222,820	1,114.1
FT-RMT-QED	@ end of computation	230,776	1,153.9
FT-RMT-QED	@ HMAC calculation	158,015	790.1
FT-RMT-QED	@ AES expand key operation	173,843	869.22
FT-RMT-QED	@ AES encryption (block 1)	197,524	987.6
FT-RMT-QED	@ AES encryption (block 2)	215,540	1,077.7

TABLE II: $\mathcal{O}_{M,R}$ timing results for a 32-bit ECU @ 200 MHz.

Operational Mode	Error Detection Point	Clock Cycles	Time (μs)
NFT	N/A	169,761	848.8
FT-RMT	@ end of computation	230,044	1,150.22
FT-RMT-QED	@ end of computation	238,796	1,193.98
FT-RMT-QED	@ AES expand key operation	89,214	446.07
FT-RMT-QED	@ AES decryption (block 1)	111,238	556.19
FT-RMT-QED	@ AES decryption (block 2)	134,101	670.5
FT-RMT-QED	@ AES decryption (block 3)	164,482	822.41
FT-RMT-QED	@ formatting received HMAC	182,466	912.33
FT-RMT-QED	@ HMAC calculation	230,564	1,152.82

Table II presents the timing results for a 32-bit ECU operating at 200 MHz for the $\mathcal{O}_{M,R}$ computations at the receiving CAN node. The FT approaches at the receiving CAN node also incur performance overhead as compared to the single-core NFT implementation. Results reveal that the FT-RMT and FT-RMT-QED incur performance overheads of 36% and 41%, respectively, at the receiving CAN node. Results verify that the FT-RMT-QED enables early detection of errors as compared to the FT-RMT for the $\mathcal{O}_{M,R}$ computations. For example, the FT-RMT-QED detects an error 140,830 clock cycles earlier (an improvement of 2.6x) as compared to the FT-RMT when the error occurs at the AES expand key operation.

B. QoS and Behavioral Reliability

An SBW system is sensitive to the delay from the driver's request at the hand wheel to the corresponding response from the front axle actuators. This delay is perceived as the QoS and can impact availability and safety if exceeded beyond a certain critical response time threshold T_{max}. The pure delay for a stable SBW system must be less than the critical pure delay T^{p}_{max} despite re-computations (permitted by FT approaches such as FT-RMT and FT-RMT-QED on error detection) and retransmissions to mask off computation and transmission errors, respectively (Section IV-B).

TABLE III: The maximum number of allowed computational runs at HW ECU1 $n1$ to yield correct result for the FAC function with T^{p}_{max} = 8 ms.

$n2$ & $n3$	$n1$ FT_a	$n1$ FT_b	$n1$ FT_c	$n1$ FT_d	$n1$ FT_e	$n1$ FT_f
$n2$ = 1, $n3$ = 1	5	5	7	6	5	5
$n2$ = 1, $n3$ = 1.167	5	5	7	6	5	5
$n2$ = 1, $n3$ = 1.833	4	4	6	5	5	4
$n2$ = 1, $n3$ = 2	4	4	6	5	5	4

TABLE IV: The maximum number of allowed computational runs at FAA ECU1 $n2$ to yield correct result for the front axle control function with T^{p}_{max} = 8 ms.

$n1$ & $n3$	$n2$ FT_a	$n2$ FT_b	$n2$ FT_g	$n2$ FT_h	$n2$ FT_i	$n2$ FT_j
$n1$ = 1, $n3$ = 1	5	5	12	9	8	6
$n1$ = 1, $n3$ = 1.167	5	5	11	9	8	6
$n1$ = 1, $n3$ = 1.833	4	4	10	8	7	6
$n1$ = 1, $n3$ = 2	4	4	10	8	7	6

We conduct experiments to determine the maximum number of allowed re-computations at SBW ECUs to yield error-free results subject to the critical pure delay T^{p}_{max} = 8 ms and $T^{hw1-faa1}_{channel}$ = 0.737 ms, which is obtained from Vector CANoe simulations [17]. The number of faults tolerated at FAA ECU1 is given by $n2 - 1$. Table III depicts the maximum number of allowed re-computations at HW ECU1 to yield correct result for the FAC function when T^{p}_{max} = 8 ms and $T^{hw1-faa1}_{channel}$ = 0.737 ms, which is obtained from Vector CANoe simulations [17]. $n2$ denotes the number of computational runs at FAA ECU1 and $n3$ denotes the number of transmissions required for error-free transmission of the encrypted message (Section IV-B). FT_a denotes FT-RMT, FT_b denotes FT-RMT-QED with error detected at the end of computation, FT_c denotes FT-RMT-QED with error detected at HMAC calculation, FT_d denotes FT-RMT-QED with error detected at AES expand key operation, FT_e denotes FT-RMT-QED with error detected at AES encryption of block 1, and FT_f denotes FT-RMT-QED with error detected at AES encryption of block 2.

Table IV depicts the maximum number of allowed re-computations at FAA ECU1 to yield correct result for the FAC function when T^{p}_{max} = 8 ms and $T^{hw1-faa1}_{channel}$ = 0.737 ms, which is obtained from Vector CANoe simulations [17]. $n1$ denotes the number of computational runs at HW ECU1 and $n3$ denotes the number of transmissions required for error-free transmission of the encrypted message (Section IV-B). FT_a denotes FT-RMT, FT_b denotes FT-RMT-QED with error detected at the end of computation, FT_g denotes FT-RMT-QED with error detected at AES expand key operation, FT_h denotes FT-RMT-QED with error detected at AES decryption of block 1, FT_i denotes FT-RMT-QED with error detected at AES decryption of block 2, and FT_j denotes FT-RMT-QED with error detected at AES decryption of block 3.

Results indicate that the FT-RMT-QED permits more (or at least equal) re-computations than the FT-RMT, depending on the error detection point at FAA ECU1, to yield correct result within the time constraints imposed by the desired QoS. For example, when $n1$ = 1, $n3$ = 2, and T^{p}_{max} = 8 ms, the FT-RMT-QED tolerates 150% more faults ($n2 - 1$) than the FT-RMT (10 allowed re-computations for the FT-RMT-QED as compared to 4 for the FT-RMT) when the FT-RMT-QED detects an error at AES expand key operation.

C. Feasibility and Scalability Analysis

Feasibility analysis of the SBW system determines if the system's end-to-end response time and pure delay are within the hard real-time constraints imposed by the desired QoS. Scalability analysis assesses system's performance for a projected addition of new components/messages later in the design process. Our scalability analysis assists cybercar's early design phases where tactical decisions such as message priority assignments are to be made in the presence of incomplete and estimated information such as bus load. To investigate the scalability of our secure and dependable SBW system over CAN, we measure the T_P and T_{res} both for the FAC function and the HWF function (T_P^{FAC} and T_P^{HWF} denote the pure delay for the FAC function and the HWF function, respectively). First, we analyze the system feasibility without any additional messages on the CAN bus. After feasibility analysis, we study the effect of additional messages on the CAN bus. For comprehensive scalability study of the SBW system, we add messages both with higher priority as well as lower priority than the SBW application's messages on the CAN bus (in the rest of this paper, we denote these messages as high-priority and low-priority for brevity).

Feasibility Analysis: To investigate the feasibility of CAN for safety-critical real-time constrained cybercar applications, we simulate our SBW system with no additional messages on the CAN bus carrying the SBW application's messages. We assume FT-RMT-QED for computations at SBW ECUs and accordingly take $T_{ecu-hw1}$ and $T_{ecu-faa2}$ to be 1.15 ms (Table I), and $T_{ecu-faa1}$ and $T_{ecu-hw2}$ to be 1.19 ms (Table II). We observe response times, pure delays, and channel times both for the FAC function and the HWF function for the SBW system with FT-RMT-QED over the run of the SBW application to capture the overall timing behavior of the SBW system. Vector CANoe measurements for our SBW system indicate the channel time, pure delay, and response time for the FAC function to be 0.737 ms, 3.077 ms, and 6.457 ms, respectively, over the entire run of the SBW application. The channel time, pure delay, and response time for the HWF function varies over the run of the SBW application, i.e., the channel time for the HWF function varies between 0.718 ms and 1.455 ms, the pure delay varies between 3.058 ms and 3.795 ms, and the response time varies between 6.438 ms and 7.175 ms. However, these delay delay variations are still well within T_{max}. The variations in channel time for the HWF function as compared to no variations for the FAC function are due to the lower priority of the HWF function's messages than the FAC function's messages. Results verify that the pure delay for both the FAC and the HWF functions are much less than the typical critical delay for an SBW system (of the order of few milliseconds to tens of milliseconds). These results establish the feasibility of our secure and dependable SBW system implementation over the CAN bus.

Effect of Additional Messages & Bus Load: Our experiments with high-priority and low-priority messages on the CAN bus

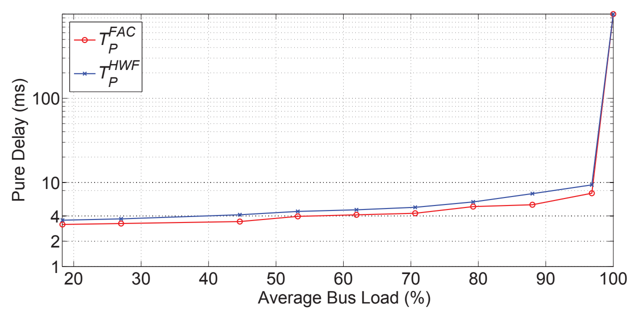


Fig. 3: Effect of increasing bus load due to high-priority messages on pure delay of the SBW application.

reveal that additional lower priority messages than the SBW application's messages have negligible effect on pure delay of the SBW application. However, higher priority messages can drastically impact pure delay of the SBW application.

Fig. 3 depicts the effect of increasing bus load due to high-priority messages on the average pure delay for the SBW application. Results show that both the average T_P^{FAC} and T_P^{HWF} increase as the bus load grows. Results indicate that T_P^{FAC} and T_P^{HWF} increases by 58.4% and 78%, respectively, as the average bus load due to high-priority messages increases from 45% to 88%. Fig. 3 shows that this delay increase is not linear and shoots up drastically above the average bus load of 97%. Results reveal that average bus load due to high-priority messages should be less than 88% for a stable SBW system that conforms to the QoS constraints imposed by $T_{max}^p = 8$ ms.

Scalability analysis suggests that safety-critical and time-constrained systems such as SBW incorporating security and dependability primitives can be implemented over CAN with careful selection of security and FT approaches as well as prudent priority assignment of SBW application's messages. In particular, for a stable SBW system over CAN, the bus load due to higher priority messages than the SBW application's messages needs to be monitored and controlled. One way of message monitoring and control is to implement firewall and authentication services in gateway nodes that connect different automotive functional domains.

VI. Conclusions

In this paper, we provide an integrated approach for the design of secure and dependable cybercars focusing on steer-by-wire (SBW) over controller area network (CAN) as a case study. The challenge is to embed both security and dependability over CAN while ensuring that the real-time constraints of the cybercar applications are not violated. Our design leverages dual-core electronic control units (ECUs) to provide fault-tolerance (FT) to the system by redundant multi-threading (FT-RMT) and FT-RMT with quick error detection (FT-RMT-QED). Results reveal that the FT-RMT-QED can detect errors maximally 140,830 clock cycles earlier (a maximum improvement of 2.6x) as compared to the FT-RMT. We quantify the number of computational errors permitted by the FT-RMT and FT-RMT-QED within the SBW system's worst-case response time threshold imposed by the desired quality of service (QoS) and behavioral reliability. Results show that the FT-RMT-QED can tolerate 150% more faults than the FT-RMT within the time constraints imposed by the desired QoS. Results verify the feasibility of our secure and dependable SBW system implementation over CAN. Scalability analysis suggests that for a stable SBW system over CAN, the bus load due to additional high-priority messages must be restricted to less than 88%.

Acknowledgments

This work was supported by the Office of Naval Research (ONR R17460) and SRC GRC Freescale grant (R65000). Any opinions, findings, and conclusions or recommendations expressed in this material are those of the author(s) and do not necessarily reflect the views of the ONR and the SRC GRC.

References

[1] ISO26262, "Road vehicles – Functional safety," April 2013. [Online]. Available: http://www.iso.org/iso/catalogue_detail?csnumber=43464

[2] C. Wilwert, N. Navet, Y.-Q. Song, and F. Simonot-Lion, *Design of Automotive X-by-Wire Systems*. The Industrial Communication Technology Handbook CRC Press, 2005.

[3] K. Koscher, A. Czeskis, F. Roesner, S. Patel, and T. Kohno, "Experimental Security Analysis of a Modern Automobile," in *Proc. of IEEE Symposium on Security and Privacy*, May 2010.

[4] C. Wilwert, Y.-Q. Song, F. Simonot-Lion, Loria-Trio, and T. Clément, "Evaluating Quality of Service and Behavioral Reliability of Steer-by-Wire Systems," in *Proc. of IEEE ETFA*, September 2003.

[5] M. L. Chávez, C. H. Rosete, and F. R. Henríquez, "Achieving Confidentiality Security Service for CAN," in *Proc. of IEEE International Conference on Electronics, Communications, and Computers (CONIELECOMP)*, March 2005.

[6] T. Hong, Y. Li, S.-B. Park, D. Mui, D. Lin, Z. A. Kaleq, N. Hakim, H. Naeimi, D. S. Gardner, and S. Mitra, "QED: Quick Error Detection Tests for Effective Post-Silicon Validation," in *Proc. of ITC*, November 2010.

[7] S. Checkoway, D. McCoy, B. Kantor, D. Anderson, H. Shacham, S. Savage, K. Koscher, A. Czeskis, F. Roesner, and T. Kohno, "Comprehensive Experimental Analyses of Automotive Attack Surfaces," in *Proc. of the 20th USENIX conference on Security (SEC)*, August 2011.

[8] M. Baleani, A. Ferrari, L. Mangeruca, A. Sangiovanni-Vincentelli, M. Peri, and S. Pezzini, "Fault-Tolerant Platforms for Automotive Safety-Critical Applications," in *Proc. of ACM CASES*, October-November 2003.

[9] F. Salewski and S. Kowalewski, "Hardware/Software Design Considerations for Automotive Embedded Systems," *IEEE Trans. on Industrial Informatics*, vol. 4, no. 3, pp. 156–163, August 2008.

[10] H. Zeng, M. D. Natale, P. Giusto, and A. Sangiovanni-Vincentelli, "Using Statistical Methods to Compute the Probability Distribution of Message Response Time in Controller Area Network," *IEEE Trans. on Industrial Informatics*, vol. 6, no. 4, pp. 678–691, November 2010.

[11] H. Schweppe, T. Gendrullis, M. S. Idress, Y. Roudier, B. Weyl, and M. Wolf, "Securing Car2X Applications with Effective Hardware-Software Co-Design for Vehicular On-Board Networks," in *Proc. of Joint VDI/VW Automotive Security Conference*, Berlin, Germany, October 2011.

[12] R. Oppliger, *SSL and TLS: Theory and Practice*. Artech House, 2009.

[13] EETimes, "How secure is AES against brute force attacks?" 2013. [Online]. Available: http://en.wikipedia.org/wiki/Brute-force_attack

[14] K. Klobedanz, C. Kuznik, A. Thuy, and W. Mueller, "Timing Modeling and Analysis for AUTOSAR-Based Software Development - A Case Study," in *Proc. of DATE*, March 2010.

[15] OpenMP, "The OpenMP API Specification for Parallel Programming," November 2012. [Online]. Available: http://openmp.org/wp/

[16] Freescale, "MPC5777M: Qorivva 32-bit Multicore MCU for Powertrain," September 2013. [Online]. Available: http://www.freescale.com/webapp/sps/site/prod_summary.jsp?code=MPC5777M

[17] Vector, "ECU Development and Test with CANoe," November 2012. [Online]. Available: http://www.vector.com/vi_canoe_en.html

[18] "Time-stamp counter," November 2012. [Online]. Available: http://www.mcs.anl.gov/~kazutomo/rdtsc.html