The Reliability of Digital Forensic Data from Networked Process Control Systems

Final Report
29 Sept 2014

Award Number 2010-DN-BX-K215

The University of Tulsa

Jeremy Daily
John Hale
Mauricio Papa

ABSTRACT

A physical system that is controlled by an array of logic devices interconnected through a communications infrastructure is effectively a networked process control system. Also known as cyber-physical systems, they are ubiquitous in American society and are used in applications ranging from medical devices, to automobiles, to robotics, to power distribution grids. Often during the investigation of a crime, the digital data stored in these control systems is useful. As such, understanding the reliability of the data from these systems is needed. Since a specific study into every networked process control system would be overwhelming, the focus of specific systems is contained to the transportation industry in the form of automotive electronic control modules. Specifically, this report shows research findings on the reliability of data captured in selected passenger vehicle air bag modules. This was done by simulating a networked system in a controlled environment to assess the accuracy of the data. Furthermore, research findings regarding the forensic capture and preservation of data from heavy vehicle engine control modules is presented along with some practical recommendations for improving the forensic soundness of extracting heavy vehicle event data. In addition to the specific studies of fielded systems, a formal methods approach is presented to show mathematical strategies to assess the reliability of the digital forensic data on networked process control systems.

TABLE OF CONTENTS

EXECUTIVE SUMMARY

Modern vehicles are cyber physical systems (CPSs) that rely on networking infrastructure to convey feedback from sensors to Engine Control Modules (ECMs) and to facilitate control of actuators. Given the importance of digital event data, a thorough system characterization and in-depth analysis of CPSs is needed to understand their behavior and reliability. As such, a formal methods approach was taken to mathematically assess some basic properties of digital forensic data with a focus on understanding event data recorders on vehicles.

Event Data Recorders (EDRs) in passenger cars record crash and pre-crash data when subjected to events, which are crash or crash like accelerations. Prior EDR testing methodologies involving crash testing are expensive and difficult to reproduce to attain statistically sound conclusions. A new methodology has been developed which allows repeatable testing and mapping of the transfer function between the vehicle controller area network (CAN) data and the EDR in a low-cost, deterministic manner. The accuracy of the 2012 Honda CR-V and 2012 Honda Civic event data recorders were tested using this new two-part methodology.

First, the test vehicles were instrumented with both a Racelogic VBOX differential GPS speed measurement system and a Vector CAN Case XL data logger. The measurements from the VBOX were transmitted onto the vehicle's CAN bus that also contained messages reflecting indicated vehicle speed, brake status, accelerator pedal position, steering wheel angle, individual wheel speeds and other signals. This put the GPS speed data on the same time base as the vehicle CAN speed signal such that no additional synchronization was required. This permitted analysis of the accuracy and update rate of the vehicle speed CAN signal, which is the source for speed data used in the Event Data Recorder (EDR).

Second, a system was developed to replay the recorded CAN data to an exemplar airbag control module in the laboratory, such that the exemplar was receiving data exactly as if it were in a moving vehicle. A pneumatic fixture with a slide was built to allow the exemplar module to be accelerated to nearly 10 km/h (6 mph) and then stopped in approximately 80 msec to create a non-deployment event that met the minimum 5 mph delta-V over 150ms threshold. Actuation of the event setting fixture was computer controlled (using LabVIEW) and synchronized with the CAN replay system so that the desired test condition could be replicated precisely. The desired test conditions were replayed to the airbag control module and a series of non-deployment events were set. Each event on the EDR data was read using the Bosch Crash Data Retrieval system.

The EDR data was compared to the network inputs, and it was determined that the two byte vehicle CAN bus signal for speed was truncated to the next lower whole km/h when recorded in the EDR. Under steady state conditions the speed data was accurate within 2%. The vehicle CAN signal published new values every 0.1 seconds, and the Honda CR-V updated values every 0.1 seconds, but the Honda Civic delayed updates by as much as 0.6 seconds during hard brake events.

Another goal of this research was the development of a forensically sound method for evidence extraction from heavy truck ECMs. Information stored in ECMs can be extracted using the engine manufacturers' maintenance software in a manner that does not protect the evidence from alteration. A method that preserves the integrity of the original evidence, is faithful to the original evidence source, and is cryptographically protected was developed. This methodology is based on the extraction and replay of ECM data with extensions specific to the manufacturer's proprietary protocols. Furthermore, a cryptosystem was designed to protect the information from modification, whether accidental or malicious. The methodology was validated by extracting and then replaying the data extracted from an actual ECM. The replay method fulfills the criteria of forensic soundness and addresses some problems with current evidence handling procedures.

The development of a sound forensic method for evidence extraction from truck ECMS is relevant due to current shortcomings on currently used procedures. Each year many thousands of trucks are involved in traffic accidents. Litigation connected with these crashes can result in judgments in the millions of dollars. In recent years, this litigation has come to depend more and more heavily on electronic event data recorded on the trucks' engine control systems. This evidence includes speed records and other event data that can help accident reconstructionists determine what transpired during the event.

Like other evidence, heavy vehicle digital event data is held to a standard of forensic soundness, which is a series of principles that ensure that forensic evidence is handled in a secure manner that guards against misinterpretation and alteration, whether intentional or unintentional.

Currently, evidence is extracted from heavy truck systems using software that was not originally designed to be forensic software, and is not particularly suited to the task. While evidence can be collected using this data in a forensically sound manner, doing so requires diligence on the part of the investigator and mistakes can result in evidence being dismissed. Additionally, the current methods by which data are stored are not particularly resistant to tampering.

The evidence contained within the heavy truck ECMs is extracted using diagnostic software provided by the manufacturer (such as CatET for Caterpillar and Cummins PowerSpec for Cummins). While it may be tempting to trust the interpretation of the OEM software, research has shown that the interpretation of the data may be an issue. For example, in some Cummins Sudden Deceleration events, speed data is reported in the report at one sample per second; however, after comparing to external reference measurements on some ECMs, researchers discovered that the data actually is reported at 0.2 second intervals (5 Hz) [1].

The proposed solution for sound forensic data extraction seamlessly integrates with the two major communications network standards used in heavy vehicles: SAE J1708/1587 and SAE J1939. These networks carry an abundant amount of information. Engine control computers monitor vehicle speed and enforce pre-set speed limits. They also broadcast information for other modules to use. For

example, speedometer and tachometer displays on the instrument panel display information received from the vehicle network. Telematics systems monitor vehicle performance and behavior using the same data. Additionally, vehicle diagnostic communications use this network. As crash evidence is extracted using diagnostic programs, vehicle networks are of interest to anyone interested in extracting crash information.

Engine control modules communicate over vehicle-specific networks, so diagnostic software requires a device that can mediate between vehicle networks and interfaces on desktop computers, such as RS-232 or USB. Those devices are known as Diagnostic Link Connectors (DLCs). They connect to a computer via a common interface and translate data to and from J1708, J1939, CAN, and other standards. During normal operation, a DLC is connected to the in-vehicle network via a diagnostic port in the cab, either an older 6-pin or a newer 9-pin SAE J1939-13 connector, also known as a Deutsch connector.

The standardization of heavy truck communication has allowed for enhanced interoperation between different brands of products, and DLC standardization is no different. All DLC manufacturers' drivers generally comply with an interface standard dictated by the American Trucking Association's (ATA) Technical Maintenance Council (TMC), known as TMC RP1210. The RP1210 interface specifies a number of functions that compliant drivers must export, including function names and formats for arguments. This allows a diagnostic program to utilize any device from any manufacturer.

A solution to problems associated with current practices is based on solutions used in the world of computer forensics. In a computer forensic investigation, a low-level copy of the evidence drive is made; this copy is known as the ``disk image.'' Forensic analysis is performed on this disk image instead of the original disk, so as to alter the original source evidence as little as possible.

This concept is extended to truck ECMs: the information would be extracted exactly one time, and then replayed for further analysis. The replay traffic information would be stored securely to prevent malicious tampering or accidental alteration. A set of four software requirements for such a solution was developed and then implemented. In particular software solutions must have the ability to: (i) record the original information extraction process, (ii) decipher any encryption or obfuscation mechanisms obscuring ECM data, (iii) store the evidence in a secure manner and (iv) respond to information requests identically to the ECM.

Implementation of such a solution required hardware platform that satisfied the following requirements: (i) relatively inexpensive, (ii) light and portable, (iii) sufficiently powerful computing resources and (iv) capable of communicating using heavy vehicle protocols.

The developed forensic extraction methodology was implemented using a replay mechanism hardware built around a BeagleBone, a commercially available ARM-based miniature computer produced by Texas Instruments. The most recent iteration of the design, the BeagleBone Black, retails for roughly $55 and has a 1GHz

ARM processor, 512MB of RAM, and 4GB of onboard flash storage. Weighing just a few ounces, it meets cost-effectiveness and portability requirements. In addition, one of the main reasons for adopting the BeagleBone platform is the capability to add functionality using expansion boards, known as Capes. Commercially available capes include a RS-485 cape and a CAN cape, supporting the physical layers of J1708/J1587 and J1939 with little to no modification.

The methodology and the platform used for its implementation were tested and validated against ECMs manufactured by Caterpillar, Inc. This manufacturer was chosen for several reasons. Firstly, it is one of the major manufacturers in the domain and secondly, their proprietary communications protocol used is encrypted in such a way that implementing a replay mechanism was particularly challenging.

In order to ensure that the method of information extraction was reliable, replays from the ECM were tested against actual extractions from CAT ECMs. The ECMs tested were evidence ECMs that were already used, and thus had been pre-populated with data. The verification procedure was designed to determine the fidelity of the information replayed, as well as the consistency of multiple replays of recorded data as compared to the consistency of multiple downloads of an evidence ECM. Using this procedure, we were able to validate and implement the proposed methodology.

1 Introduction

1.1 Cyber Physical Systems Analysis

Modern vehicles are cyber physical systems (CPSs) that rely on networking infrastructure to convey feedback from sensors to Engine Control Modules (ECMs) and to facilitate control of actuators. Given the importance of safety controls, thorough system characterization and in-depth analysis of CPSs is needed to understand their behavior and to prevent unexpected and potentially dangerous side effects. Unfortunately, analysis is oftentimes hampered by CPSs' proprietary designs that make accurate system characterization very difficult, if not impossible. Needed are strategies for conducting a full range of analyses over vehicular CPS elements, spanning experimentation, simulation and formal verification.

1.1.1 Experimental Platforms for Real Time Analysis

A primary challenge in characterizing and analyzing parts of a vehicle's CAN system to understand the behavior of specific ECUs and evaluate operational boundaries, is access to the system. Preserving the real-time characteristics of the data transmission as well as ensuring experimental repeatability is very difficult when driving the car to generate data sets. One solution is to record the data generated during one test drive and design a process to replay that data in real-time with high temporal accuracy of the original message transmission timing. A straightforward approach would be using a general purpose PC and a suitable programming environment, e.g. C, C++ or Matlab, to recreate the recorded data transmission.

However, operating systems used on general purpose computers do not provide guarantees on timeliness of program execution. Thus, the relative execution timing of a real-time analysis might vary depending on other processes running on the system. Alternative platforms for real time analysis therefore must be considered.

A real-time operating (RTOS) is a special-purpose operating system that imposes rigid time requirements on its process execution and enables the design of real-time applications. A hard real-time system guarantees that all delays within the system are bounded by an upper and a lower execution time that must be met at all times. A soft real-time system does have upper and lower bounds for all functions but it assigns and manages varying levels of task priorities.

Field-programmable Gate Arrays (FPGAs) offer a programmable hardware layer made up of configurable logic blocks (CLBs), block random access memory (BRAMs) and digital signal processing (DSP) blocks. FPGAs can be programmed via a hardware description language such as Verilog or VHDL. Manufacturers have started to investigate the possibility of using OpenCL to program FPGAs. FPGAs allow several programs to execute truly in parallel without competition for shared resources and may offer down to nanosecond response times for input-to-output processing.

1.1.2 Formal Verification

Hybrid automata are formal models of CPSs, linking agent behavior in discrete and continuous domains [2]. Henzinger identifies certain classes of automata, specifically linear hybrid automata, which lend themselves to the fully automatic application of various model-checking techniques. Indeed, to demonstrate the potential of their approach, Henzinger et al. developed the symbolic model checker HyTech for linear hybrid automata [3]. Hybrid programs are structured control programs that can capture the discrete and continuous transitions of hybrid system. It has been shown that the expressive power of hybrid programs is suitable to represent hybrid system descriptions formulated as hybrid automata. The tool Keymaera is a theorem prover that supports the definition of hybrid systems in form of hybrid programs [4]. Platzer introduced Keymaera's specification and verification language, the Dynamic Differential Logic (dL) and demonstrated its utility by modeling and analyzing certain parts of the European Train Control System.

1.2 Event data recorders in passenger cars

1.2.1 Motivation for EDR studies

Event Data Recorders (EDRs) have been equipped in select cars for many years and are becoming more pervasive in over the road vehicles. Originally, EDRs were developed as a tool to help automotive engineers understand crash dynamics in order to make cars safer. For example, in 1992 General Motors (GM) installed crash data recorders in Indy race cars. These devices provided information on the human body's tolerance to impact and velocity change; information which helped improve the safety of both racing and passenger cars. In 1994, GM implemented recording

capable sensing diagnostic and modules (SDMs) to select passenger cars. These devices recorded the change in longitudinal velocity (delta-v), allowing GM engineers to study and improve the restraint system of their vehicles. By 1999 certain GM SDMs were able to record pre-crash data such as engine speed, vehicle speed, brake switch status, and throttle pedal position. EDRs have continued to develop over the years are now present in many cars. Most vehicle EDRs offer more crash and pre-crash data than the GM 1999 SDMs. Although these devices were originally intended for manufacturer studies to increase the safety of their automobiles, EDR data is interesting to crash investigators and insurance companies.

EDR data is used by accident reconstructionists and law enforcement to help determine events leading to a crash. EDR data was used in criminal court in the 2002 *Colorado vs. Cain* case. Since then, EDR data has been used in over 19 state courts and at the federal level.

The use of EDR data in court cases as scientific evidence places requirements on EDR data and its validity. As established in Daubert, scientific data presented by an expert witness is subject to review and must be "sufficiently established to have general acceptance in the field to which it belongs." The Daubert standard for admissibility of scientific evidence requires that EDR data be verified using an accepted peer reviewed method. Although methods exist for the analysis of EDR data, these methods can be improved, particularly in reproducibility. Current methods limit the ability to quantify data error ranges in a statistically significant manner.

EDR data became standardized in passenger vehicles in September of 2012 through part 563 of the NHTSA (National Highway Traffic Safety Administration) 49 Code of Federal Regulations (CFR) ruling, which gives a minimum data set required if the vehicle is equipped with an EDR [5]. Tables showing the minimum required are given in Appendix A. Within the part 563 ruling the following standards have been established for pre-crash data:

Table 1: Minimum EDR Data Required by NHTSA CFR 49 Part 563

Data Element Name	Condition for Requirement	Recording Time Interval (relative to time zero)	Data Sampling Rate (per second)
Speed, Vehicle Indicated	Required	-5 to 0	2
Engine Throttle (% full) Accel Pedal Position (%full)	Required	-5 to 0	2
Service Brake (on/off)	Required	-5 to 0	2
Engine Speed	If recorded	-5 to 0	2

(RPM)			
ABS Activity (engagement/non-engagement)	If recorded	-5 to 0	2
Stability Control (on, off, engaged)	If recorded	-5 to 0	2
Steering Input (degrees)	If recorded	-5 to 0	2

CFR 49, part 563, specifies that all manufactures must make EDR data available if EDR data is recorded, whether that be through a dealership scanning tool or a third party tool. The Bosch Crash Data Retrieval kit (CDR) is one such third party tool. This device supports hundreds of cars and is able to harvest the EDR data from a module. The Bosch CDR tool was used in this study.

Prior EDR testing methodologies required setting events in the airbag control module of the vehicle during controlled driving. Duplicating events was nearly impossible and it was difficult to determine differences in recorded speeds to reference speeds based on measurement error, wheel slip, reporting time delays, or data truncation within the EDR. Recording thresholds may have increased making non-deployment and deployment events closer in magnitude, which increase the risk of accidentally exceeding the deployment threshold while setting events. Because of the shortcomings of existing methods of EDR data analysis a new method of assessing the accuracy of EDR data was developed and is the subject of this research.

The new methodology eliminates the risk of accidentally deploying airbags while gathering external validation data and vehicle network data in the test vehicle. The techniques presented in this work also allow data gathering without tampering with the airbag control module, which reduces the potential liability to testers using rental or borrowed test vehicles. The new methodology allows for repeatable testing and mapping the transfer function between the vehicle CAN bus data and the EDR data. Should a manufacturer make a design change to an EDR, identical inputs can be given to the new EDR and changes in its behavior can be documented. This methodology allows researchers the ability to re-create events of interest in a low-cost, repeatable manner. As an example, the accuracy of the 2012 Honda CR-V and 2012 Honda Civic event data recorders were tested using this new two-part methodology.

1.2.2 Literature Review

A large portion of this literature review is taken from a 2013 SAE World Congress publication by Diacon et al. [6].

In 1983 Bosch began development of a networking system for cars, namely CAN (controller area network). At the 1986 SAE World Congress Bosch presented CAN and subsequently released all intellectual property concerning it, which resulted in

a drop in costs for its implementation [7]. Since 2008, most vehicles have implemented some version of a CAN for on-board device communication. To briefly describe the network of CAN itself; CAN is a multi-master broadcast serial bus with a non-return to zero (NRZ) bit encoding and automatic collision detection and message arbitration. Mueller, et al. presented a method for evaluating the accuracy of CAN messages in Ref. [8]. In this paper CAN histories were compared to professional grade measurements, allowing quantification of the accuracy of CAN messages such as speed, and thus an analysis of EDR data. This means that some of the EDR data depends on the CAN bus data.

The Bosch CDR tool help file and the Data Limitations section in CDR reports contain useful information about accuracy limitations and EDR transfer functions. A section of the 2012 Honda Data Limitations section is shown verbatim below.

The Bosch CDR tool help file and the Data Limitations section in CDR reports contain useful information about accuracy limitations and EDR transfer functions. A section of the 2012 Honda Data Limitations section is shown verbatim below.

"**Data Limitations"**

General Information:
These limitations are intended to assist you in reading the event data that has been imaged from the vehicle's SRS control unit. They are not intended to provide specific information regarding the interpretation of this data. Event data should be considered in conjunction with other available physical evidence from the vehicle and scene.

Honda and Acura passenger vehicles designated as 2013 or later model year production are designed to be compatible with the Bosch CDR tool. However, due to production variations during the 2012 model year, only certain 2012 model year vehicles are compatible with the Bosch CDR tool.

Recorded Crash Events:
Data for front, side, rear and rollover events can be recorded as either non-deployment or deployment events. Both types of events can contain precrash and crash data.
- A non-deployment event is recorded if the change in longitudinal or lateral velocity equals or exceeds 8km/h over a 150ms timeframe or another type of non-reversible deployable restraint device other than a front, side, or side curtain airbag (e.g. seatbelt pretensioner) is commanded to deploy. Except as indicated below, non-deployment events are not locked into memory and can be over-written by subsequent non-deployment or deployment events.
- A deployment event is recorded if front airbag(s), side airbag(s), or side curtain airbag(s) are commanded to deploy. Deployment events are locked into memory and cannot be over-written.
The SRS control unit typically records only one event. Two events can be recorded if the T0 (time zero) values for each event occur within 5 seconds of each other. T0 is established by whichever of the following occurs first: (1) the change in longitudinal velocity at the SRS control unit equals or exceeds 0.8km/h over a 20ms timeframe; (2) the change in lateral velocity at the SRS control unit equals or exceeds 0.8km/h over a 5ms timeframe; or (3) a commanded deployment of any type of non-reversible deployable restraint device (e.g. airbag or seatbelt pretensioner). Therefore, a non-deployment event can be recorded and locked if it occurs within 5 seconds of a deployment event.

Data:
- Data recorded by the SRS control unit and imaged by the CDR tool is displayed relative to T0, not the time at which the vehicle made contact with another vehicle or object.
- Pre-crash data is recorded at 2 samples per second starting 5 seconds before T0.
- Crash data is recorded at 100 samples per second from T0 to 250 milliseconds or T0 to TEnd (end of event) plus 30 milliseconds, whichever is shorter. TEnd occurs when the change in longitudinal and lateral velocity equals or falls below 0.8km/h over a 20ms timeframe.
- All data is displayed in SAE J211 sign convention unless otherwise noted in this document.
- Delta V, longitudinal reflects the change in velocity that the SRS control unit experienced in the longitudinal direction during the recorded portion of the event and is not the speed the vehicle was traveling before the event.
- Depending on the severity of the event and the accelerometer characteristics, saturation of the SRS control unit longitudinal or lateral accelerometers may occur, decreasing the recorded Delta V value.

- Speed, vehicle indicated data accuracy can be affected by various factors, including but not limited to the following:
- Significant changes in tire size from the factory setting
- Wheel lockup
- Accelerator pedal position, percent full is the ratio of accelerator pedal position compared to the fully depressed position.
- PCM (Powertrain Control Module) derived accelerator pedal position, percent full may differ from the accelerator pedal position, percent full under circumstances such as brake override activation or cruise control system engagement. These circumstances are based on vehicle equipment application and vary by model.
- Steering input angle is recorded in 5 degree increments (e.g. if actual steering input = 13.4 degrees, recorded value would be = 15 degrees).
- Side air bag suppression system status, right front passenger is recorded when the vehicle is equipped with the Occupant Position Detection
System (OPDS).
- Occupant size classification, right front passenger airbag suppressed data is recorded as yes (suppressed) if the front passenger seat weight sensor system determined the passenger seat was empty or occupied by a child-size occupant.
- If power to the SRS control unit is lost during an event, all or part of the data may not be recorded

While this data is helpful, actual test data is needed to verify and quantify their claims, specifically concerning the issues of data resolution and truncation. Many EDR accuracy studies have been conducted and are summarized below.

EDR pre-crash speed data was first recorded by General Motors beginning in some 1999 models and the first paper to address pre-crash EDR speed data accuracy was by Chidester in 1999 [9]. Chidester's team included General Motors personnel, and the paper listed the accuracy of speed data as +/-4% of the recorded value, implying that differences between EDR recorded values and ground speed may be higher at greater speeds. This paper was written before the start of 1999 model-year production, and the 4% articulated a design tolerance that could include tread wear, tire pressure variations, and other design variations. Test data was not yet available. The paper noted that the various data elements were recorded asynchronously, raising the issue that the timing labels associated with pre-crash data may not be precise regarding when the data was actually recorded. Data points labeled "-1" may actually be recorded any time during the second before algorithm enable (AE).

In 2003, Lawrence [10] created artificial crash signals during normal driving and published data finding the GM EDR speed to be under reported by 1.5 km/h (about 1 mph) at low speeds and over reported by 3.7 km/h (about 2.3 mph) at high speed when compared to reference instrumentation, under steady state conditions. This paper raised the concern that differences between recorded speed values and speeds obtained with external instruments may not only have some relationship to vehicle speed, but that there may be some type of offset since the sign of the difference changed between low speed and high speed.

In 2005, Niehoff reported the recorded pre-crash speed from 28 NHTSA crash tests of GM and Toyota vehicles [11]. The crash tests were at speeds of 48 and 64 km/h (30 to 40 mph) and the data was reported as “within 1 mph”. The pre-crash EDR data was typically reported in whole miles per hour and the reference instrumentation was reported with either one decimal place or no decimal places. Data was not reported as a percentage of speed, which was appropriate given the combination of resolution of the data and the relatively low speed of the test.

In 2006, Wilkinson reported on the timing of EDR data in General Motors sensing and diagnostic modules, indicating that the actual time between labeled data points may vary from the interval suggested by the labels [12]. In 2011, Bare et al. reported on pre-crash data timing of the GM SDM-DS module, indicating that the timing of the last data point recorded can vary from the label "-1" that is placed on that last data point [13]. This work addresses some of the same type of timing issues as Wilkinson but used more sophisticated instrumentation and was on a more recently designed EDR. Bare also suggested that data timing was far closer to the reported "1 second" interval than reported by Wilkinson.

In 2008, Gabler [14] reported on the accuracy of pre-crash speed data for 33 crash tests ranging from 40 to 56 km/h (25 to 40 mph) on 2004-2007 model year vehicles. 32 of the tests were on GM vehicles and one was on a Toyota. The paper states that all of the speed data was within 3%, except for test 5310 on a 2005 Buick Rendezvous that reported low by 22% (27 mph vs. 35mph actual). Gabler did not explain the anomaly.

In 2008, Ruth [15] reported on the steady state speed data accuracy of Ford Powertrain Control Module event data recorders at speeds from 48 km/h to 113 km/h (30 to 70 mph). For the 2005 Ford Crown Victoria, the data was accurate within 1.0%. Similarly, Ruth et al. reported on Ford air bag control modules in Ref. [16]

Takubo and Ishikawa et al. reported on Japanese New Car Assessment Program (NCAP) tests and additional tests intended to mimic real world crashes [17] [18]. They published two SAE papers with each successive paper including some additional tests. As such, the most complete data set as of this writing can be found in the latest, though the earlier papers contain some minor details which are not reproduced in the most recent paper. They reported that the cars were mostly 2007 and 2008 Toyota Corollas (00/02 EDR), and that “the pre-crash velocities [reported by the] EDR were highly accurate and reliable but generally lower than the optically derived velocities.” In 14 full overlap barrier tests the EDR speed data was ‐2.0% different than the reference instrumentation, with a range of ‐6.3 to +1.9%, and an RMS of 2.6%. For the 14 Offset Deformable Barrier (ODB) tests, the EDR averaged ‐2.1% different than the reference instrumentation, with a range of ‐4.1% to 0.0% and an RMS of 2.7%. The negative average value is consistent with the CDR Data Limitations, which state the speed data is truncated to the next lower even km/h value.

In 2009, Ruth [19] reported on the speed data accuracy of Chrysler vehicles. In 113 km/h steady state conditions the 2008 Jeep Commander EDR reported from -1.18 to +0.32 km/h different than GPS, and Dodge Dakota EDR reported from -3.09 to -0.98 km/h different from the GPS reference instrumentation, with the average error being below zero due to truncation of any fractional km/h to the next lower whole number in the EDR.

In 2010, Bortolin [20] reported that a 2008 Dodge Caravan EDR reported from -1.74 to +0.63 mph of GPS reference speeds from 11 to 61 mph

In 2010, Ruth reported on the accuracy of the 2009 Ford Crown Victoria Powertrain Control Module (PCM) EDR in steady state and heavy braking [21]. The Crown Victoria vehicle speed sensor is on the transmission output shaft, and during heavy ABS controlled braking wheel slip results in the speed being under reported by an average of 5% at 97 km/h. This recognized that the "vehicle indicated speed" on the speedometer or CAN bus obtained from a sensor measuring proportional to wheel speed might not represent the ground speed of the vehicle under heavy braking conditions.

In 2011, NHTSA conducted an evaluation of Toyota pre-crash data accuracy [22]. The paper reports 28 staged events using two 2007 Camry's (04/06 EDR) and a 2008 Highlander (04/06 EDR) as bullet vehicles striking the back of a 2006 Tacoma (00/02 EDR) target vehicle with a 3 to 8 km/h (2-5 mph) closing speed in order to create a non-deployment event the EDR would capture. NHTSA defined the vehicle speed tolerance as +/- 2.3 km/h (1.5 mph), and was aware that the recorder temporary buffer only refreshed speed data every 0.5 seconds. Around each EDR data point they created a "window of acceptance" of +/- 2.3 km/h (1.5 mph) that extended back in time 0.5 seconds. If the GPS speed data crossed anywhere in the window, then it was deemed within the acceptable tolerance. NHTSA concluded that 100% of the pre-crash speed data fell within the tolerance and time window. The +/-2.3 km/h window was wider than the +0/-2 km/h range expected from the data resolution cited in the CDR Data Limitations.

In 2012, Ruth et al. [23] reported on 2010 and 2011 Toyota Camry EDR speed data. Data limitations stated that the Toyota truncated speed data to the next lower even number of km/h, such that the difference between EDR and GPS would be expected to from -2.0 to 0 km/h in the absence of other calibration or random measurement errors. For steady state conditions, differences for the 2011 Camry at 113 km/h (70 mph) ranged from -3.0 to -0.4 km/h (-1.9 to -0.2 mph). For the 2010 Camry, for the 113 km/h (70 mph) tests the difference ranged from -2.1km/h to +0.2 km/h. The best fit line has a slight slope indicating a larger difference would be expected at a higher speed, but the slope was not statistically significant. During maximum ABS braking, Ruth identified that in addition to the wheel slip under reporting phenomena, data recorded for the last point prior to impact may be up to 0.5 seconds old, resulting in over reporting speed. This research reinforced that for speed at impact calculations, the timing of the last speed data point recorded was important.

To summarize the literature, EDR speed data was first published at +/-4% in 1999 in a paper specifically addressing GM recorders. Since then specific GM, Ford, Chrysler and Toyota EDR's have been tested and the data has been found to be more accurate than the previously published 4% under steady state conditions. Tests have evaluated a vehicle at a point in time, and most indicate the EDR data is on average under-reported due to some form of truncation of the raw data to the next lower integer or even integer number. Tire wear over the life of the vehicle has not been addressed since the original 1999 paper. The more recent literature also documents that under maximum ABS braking conditions, the EDR will accurately report the average drive wheel speed it is measuring, but it will under-report the true ground speed. Tests on 2010/2011 Toyotas indicated the CAN bus updated only every 0.5 seconds, and during braking the reporting delay could lead to the last speed data point being over-reported.

1.3 Forensics of heavy truck ECMs

Thousands of trucks are involved in traffic accidents each year; in 2008, 380,000 trucks were involved in accidents [24]. Litigation connected with these crashes can result in judgments in the millions of dollars. In recent years, this litigation has come to depend more and more heavily on electronic event data recorded on the trucks' engine control systems. This evidence includes speed records and other event data that can help accident reconstructionists determine what transpired during the event.

Like other evidence, heavy vehicle digital event data is held to a standard of forensic soundness, which is a series of principles that ensure that forensic evidence is handled in a secure manner that guards against misinterpretation and alteration, whether intentional or unintentional.

Currently, evidence is extracted from heavy truck systems using software that was not originally designed to be forensic software, and is not particularly suited to the task. While evidence can be collected using this data in a forensically sound manner, doing so requires diligence on the part of the investigator and mistakes can result in evidence being dismissed. Additionally, the current methods by which data are stored are not particularly resistant to tampering.

1.3.1 Truck ECMs and Digital Evidence

Beginning in the early 1990s, truck engine manufacturers implemented electronic engine control in order to meet more stringent emissions requirements placed on large diesel vehicles. Over time, demand from fleet customers lead to implementation of data logging within these engine control computers, including fuel usage tracking and driver behavior. NHTSA requested the engine manufacturers to investigate the implementation of event data recorder (EDR) functionality in heavy truck ECMs. Shortly thereafter, manufacturers began offering this functionality in their ECMs, first as add-on packages then as standard features.

Table 2: ECMs and associated software

ECM Family	*Software*
Caterpillar	Caterpillar Electronic Technician (CatET)
Cummins	Cummins PowerSpec, Cummins Insite
Detroit Diesel	DDEC Reports, Detroit Diesel Diagnostic Link (DDDL)
Mack	Proprietary to Mack and Volvo
Mercedes	DDEC Reports, Detroit Diesel Diagnostic Link (DDDL)
Navistar	ServiceMaxx
Paccar	Cummins PowerSpec, Cummins Insite, or Paccar Davie
Volvo	Mack and Volvo Proprietary

The evidence contained within the heavy truck ECMs is extracted using diagnostic software provided by the manufacturer. Some popular manufacturers and the associated software of forensic interest is listed in Table 2: ECMs and associated software.

While it may be tempting to trust the interpretation of the OEM software, research has shown that the interpretation of the data may be an issue. For example, in some Cummins Sudden Deceleration events, speed data is reported in the report at one sample per second; however, after comparing to external reference measurements, it is discovered that the data actually is reported at 0.2 second intervals (5 Hz) [1]. Similarly, Austin and Farrell [25] showed that many Snapshot records from Caterpillar are reported every 0.5 seconds instead of every second as represented in the OEM software.

1.3.2 Heavy Vehicle Networks

In modern automobiles, the number of in-vehicle electronic devices that need to communicate with one another has seen dramatic increases: engine control modules, anti-lock brake system modules, transmission control modules, collision detection systems, and even entertainment systems are examples of devices that need to communicate. This leads to a combinatorial explosion of communication links, that requires the use of a common multiple-access bus. Currently, almost all functions in automobiles are carried out over the network. Notably, in-vehicle networking has allowed external diagnostic connectors to receive diagnostic fault codes through a port in the cab.

Heavy trucks are no different. Engine control computers monitor vehicle speed and enforce pre-set speed limits. They also broadcast information for other modules to use. For example, speedometer and tachometer displays on the instrument panel display information received from the vehicle network. Telematics systems monitor vehicle performance and behavior using the same data. Additionally, vehicle diagnostic communications use this network. As crash evidence is extracted using diagnostic programs, vehicle networks are of interest to anyone interested in extracting crash information. There are two major vehicle network standards used

in heavy trucks published by the Society of Automotive Engineers, SAE J1708/1587 [26] [27] and SAE J1939 [28].

1.3.3 Diagnostic Link Connector Hardware and Software

Engine control modules communicate over vehicle-specific networks, so diagnostic software requires a device that can mediate between vehicle networks and interfaces on desktop computers, such as RS-232 or USB. Those devices are known as Diagnostic Link Connectors (DLCs). They connect to a computer via a common interface and translate data to and from J1708, J1939, CAN, and other standards. During normal operation, a DLC is connected to the in-vehicle network via a diagnostic port in the cab, either an older 6-pin or a newer 9-pin SAE J1939-13 connector, also known as a Deutsch connector.

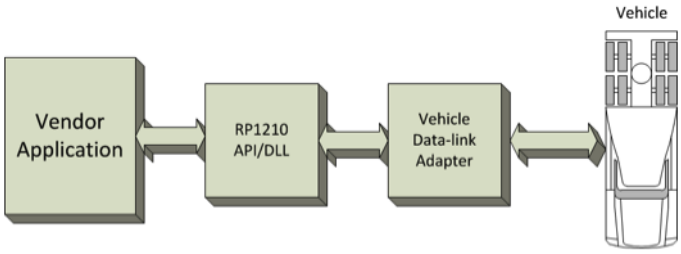


Figure 1: Diagram of RP1210 device and driver.

The standardization of heavy truck communication has allowed for enhanced interoperation between different brands of products, and DLC standardization is no different. All DLC manufacturers' drivers generally comply with an interface standard dictated by the American Trucking Association's (ATA) Technical Maintenance Council (TMC), known as TMC RP1210 [29]. The RP1210 interface specifies a number of functions that compliant drivers must export, including function names and formats for arguments. This allows a diagnostic program to utilize any device from any manufacturer (Figure 1).

1.3.4 Current Practice

Information from a heavy truck vehicle is typically downloaded using the engine manufacturer's diagnostic software. The normal use case for diagnostic software is connecting to a diagnostic port in the cab of the truck using a diagnostic link connector, and accessing the desired information with the maintenance software.

However, extracting crash information is not a normal use case. If the truck has been in a high speed collision, the electrical connection between the cab and the engine may have been severed, or the cab itself may be physically inaccessible. In these cases, the engine control module must be removed and analyzed separately. This is performed either by placing the ECM in a donor vehicle identical to the one in the crash, or by connecting to the ECM directly using the manufacturer's engine flashing harness.

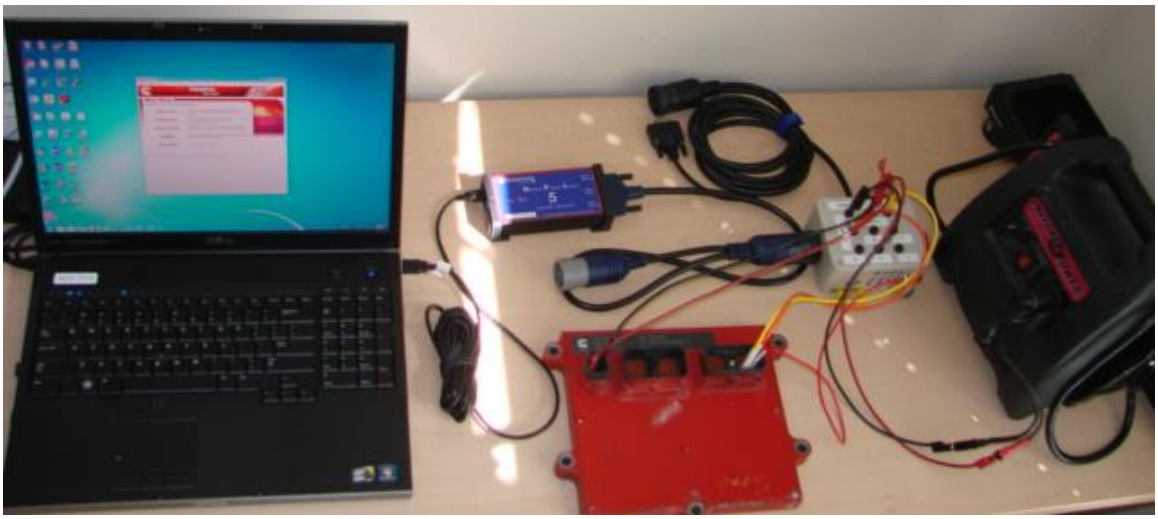


Figure 2: Bench download from a Cummins ECM.

Whatever method was used to connect to the ECM, the investigator then proceeds to extract the information using the manufacturer's diagnostic software. This includes event data from the crash itself, mechanical fault information, engine tuning parameters, and driver behavior information. This information is then recorded, in most cases, by printing the data to a PDF file. However, in many instances the software does not support printing data and so it must be captured using a screenshot. All of the files resulting from the extraction are then stored on some kind of removable media and produced to the client.

1.3.5 Forensic Soundness Requirements for ECMs

A heavy vehicle electronic control module (ECM) is a specialized process control computer that may have data of interest to an investigator. While ECMs may not have keyboards and monitors, they still are computers and have central processing units, memory, storage, and a means of networking or communicating with external devices. As such, many of the principles from computer forensics can be applied to heavy vehicle ECMs.

The first litmus test that any forensic methodology must pass is the rules of evidence for the legal system in which it is used. Here, we will focus on the laws of the United States. The two primary standards to which evidence is held in the US are the Daubert standard, established in the case *Daubert v Merrell Dow Pharmaceuticals*, and the Federal Rules of Evidence Rule 702 on expert testimony [30]. The following is a summary of the relevant criteria for the Daubert standard: (i) testimony must be relevant and reliable, (ii) Judges have the task of ensuring that expert testimony rests on a reliable foundation and is relevant and (iii) some or all of certain specific factors, including testing, peer review, error rates, and acceptability in the relevant scientific community, may prove helpful in determining reliability of forensic testimony. Similarly, this is the summary of relevant

requirements according to Federal Rules of Evidence Rule 702: (i) method can be and has been tested, (2) method has been subjected to peer review or publication and (iii) method has a known error rate

A review of the rules of evidence as they relate to digital forensics may be found in [31]. Any forensic evidence must meet these standards in order to be admissible.

In addition to the general rules of evidence, the digital forensics literature contains many attempts to quantify the concept of forensic soundness. A Special Report issued by the National Institute for Justice on Electronic Crime Scene Investigation \cite{NIJ2008} highlights the following basic forensic principles applied to dealing with digital evidence: (i) any process or procedure of collecting, transporting, or storing of digital evidence should not incur any changes to the evidence, (ii) only specifically trained experts should examine digital evidence and (iii) transparency during the operations of acquisition, transportation, and storage of the evidence should be maintained.

These basic tenets lay the foundation for the idea of forensic soundness. It is important to understand the term ``forensically sound'' as it relates to digital evidence. Many authors and professional organizations have attempted to rigorously define this concept, including the National Institute for Standards and Technology (NIST) [32], law enforcement entities such as the National Institute for Justice (NIJ) [33], and academic bodies like the International Organization on Computer Evidence [34]. The methods of extracting, analyzing, and presenting digital evidence are forensically sound if they perform the task in a manner such that the results can be used in legal proceedings with a high degree of confidence in their admissibility. We refer to the process of extracting, interpreting, and presenting evidence as the ``forensic process.''

In addition to the NIJ report, McKemmish [35] enumerates the following components of forensic soundness: (i) **meaning** (a term that denotes confidence in the interpretation of extracted evidence data), (ii) **error detection** (denotes processes for detecting or predicting errors in the forensic process), (iii) **transparency** (the forensic process is documented, known, and verifiable) and (iv) expertise is required for those investigators examining digital data.

Futhermore, Casey [36] defines 7 levels of certainty for digital evidence that dictate how much weight should be given to the evidence in a case: C0 (evidence contradicts known facts), C1 (evidence is highly questionable), C2 (only one source of evidence that is not protected against tampering), C3: (source(s) of evidence are more difficult to tamper with but there is insufficient evidence for a firm conclusion, or unexplained inconsistencies exist in available evidence, C4 (sole source evidence is protected against tampering or multiple, independent sources of evidence agree but the independent evidence is unprotected from tampering), C5 (agreement of evidence from multiple, independent sources protected from tampering, but small uncertainties exist) and C6 (the evidence is tamper proof and unquestionable.

While many of these levels include requirements that are encompassed by the requirements of error detection and certainty of meaning specified by McKemmish,

this standard includes the principle of tamper resistance. In addition to protecting the evidence from tampering, a system which can demonstrate the absence of tampering also fulfills this requirement.

1.3.6 Shortcomings of current practices and the need for a new approach

Though several authors describe the concept of forensic soundness slightly differently, the basic concepts remain the same. The evidence must be extracted in a transparent, repeatable, and verifiable manner that alters or destroys as little data as possible. There must be a high degree of certainty that the data is interpreted correctly. There must be measures taken to ensure that the evidence has not been tampered with.

Unfortunately, current practices fail to meet these standards in several respects. First and foremost is that evidence extractions cannot be relied upon to leave the data unaltered. Some engine maintenance software can, by default, clear trip data after they are extracted from the engine. While this may be desirable behavior for the software's typical use case, in a forensic context it is unacceptable. In addition, if a bench download is being performed, the ECM is disconnected from all of the various systems it was designed to monitor, causing spurious fault codes to be created. With some manufacturers' products, especially Caterpillar, these can overwrite fault codes of interest in a crash investigation. The damage is made worse when repeated downloads are conducted.

In addition to data extraction, current practices lack in the forensic soundness of the storage of data as well. No measures are taken to ensure that data have not been tampered with. Data export formats, typically plaintext HTML or PDF documents, can easily be altered with readily-available software. Some manufacturers' native file formats are encrypted somewhat, though not strongly: Cummins' EIF format is simply a password-protected ZIP archive, while Detroit Diesel's Drumroll files are encrypted using a simple XOR cipher. Once decrypted, these files can be altered to remove evidence and re-encrypted easily.

Due to the potential for destruction of the source evidence and alteration of extracted data, it appears that current practice fails to pass the litmus test of forensic soundness as currently accepted in the digital forensics community.

In reference [37], Johnson, Daily, and Kongs describe the current shortcomings of heavy truck ECM forensics, reviewing the weaknesses previously enumerated in this section. They also demonstrate a practical example of file modification of a DDEC Reports data file, altering vehicle speed records.

The literature relating to the forensics of heavy truck ECM data is relatively undeveloped. Most of the work to date, such as references [38], [39], and [25] deal with the accuracy of data recorded on ECMs and causes of data loss. While these works address the practical necessities of extracting data from ECMs and ensuring that the data are correct, they do not consider the possibility of intentional modification to data or corruption of data after it has been extracted.

2 Methods

2.1 Methods for Cyber Physical Systems

2.1.1 Logger Design and Evaluation

We evaluated three Arduino Can Logger (ACL) designs of increasing complexity and computational power. The first ACL design constitutes an Arduino UNO and a CAN bus shield. The second ACL design is based on the combination of the Arduino UNO, the CAN bus shield, and the OpenLog chip. The third ACL design consists of an Arduino Due (more powerful than the UNO), the CAN shield, and the OpenLog chip.

To evaluate the logging implementations, two methods are used 1.) FPGA Deterministic CAN Message Generator 2.) Toyota RAV 4 CAN Traffic recording. The first method utilizes the National Instruments CompactRio platform featuring an NI 9853 CAN module to send specifically timed CAN messages with predefined content. The CompactRio is connected to a CAN infrastructure that includes an industrial-strength Vector CANCaseXL CAN logger. 500 predefined CAN messages are transmitted 35 times and the logging data generated by the two logging components is then analyzed against defined performance characteristics. The second evaluation method, features the replay of recorded 2009 Toyota RAV 4 CAN traffic via the Compact Rio platform and a replay system specifically designed to enable deterministic CAN replay analysis. The CAN messages, their content, and their timing is representative of CAN traffic found in modern passenger vehicles. The replay and recording procedure is performed 35 times and analyzed as before.

2.1.2 Real time Replay Methodology

Deterministic and faithful replay behavior is vital to reproducible experimentation and testing since it permits reducing the difference in the observed time values between original data logging and replay logging. This means that if the replay algorithm produces deterministic time values in repeated replay sessions, it should be possible to reduce the timing differential that might be caused by different network topologies.

Figure 3 illustrates the Real Time Replay Methodology. Data collection is achieved by a VectorCANcaseXL CAN logger. Data transformation involves adjusting timestamps to calibrate and condition data with respect to performance characteristics of the reference CAN logger. During replay, traffic is logged and its temporal properties are compared against that of the original data. An acceptable outcome in this comparison validates subsequent analysis, otherwise adjustments are made to bring the replay in line with the original traffic and the process is repeated.

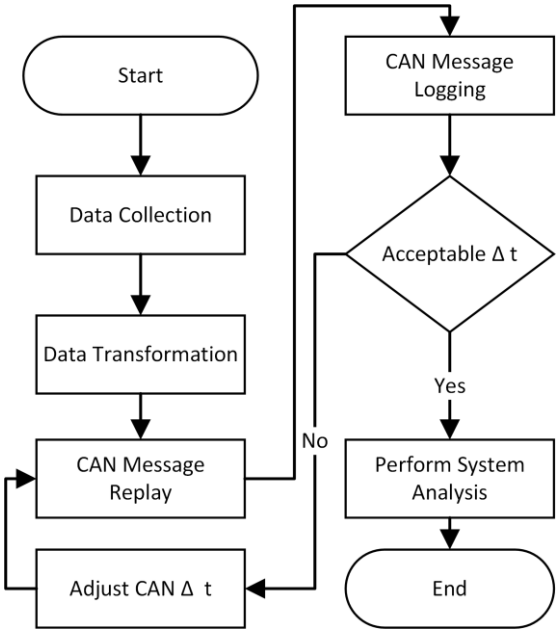


Figure 3: Real Time Replay Methodology.

2.1.3 Logger Evaluation

The Arduino-based solutions are compared against the CompactRio NI 9853 CAN logger in the context to the Real-time replay process. The CAN replay evaluation of the loggers uses the real-time replay system to send out recorded CAN traffic and to compare the logfiles to reference data stored in the database for comparison. The order and completeness criteria established before must be met perfectly by any viable logging solution. The timing criteria does not have to be met perfectly, but must be appropriate for the purpose of the cyber-physical system analysis. The evaluation framework consists of the necessary CAN wiring harness/cables, the CompactRio platform with a NI 9853 module and the respective logger under evaluation.

2.1.4 System Characterization

The real time replay methodology is used in a system characterization and analysis of a reference component on a vehicle CAN infrastructure. Specifically, a voltage sensor for a Transmission Control Unit (TCU) is selected for characterization and analysis. This process required profiling and deriving the pin out for the TCU of a 2009 VW Jetta. An internal voltage sensor measures voltage supplied to the TCU (which should not exceed 12.5 V) and transmits that information in the form of a CAN message.

To characterize the TCU supply voltage sensor and the CAN conversion process a real-time characterization system was developed that: 1.) Monitors the supply voltage provided to the TCU, 2.) queries the TCU supply voltage via OBD-II compliant messages, and 3.) provides a means to compare both measurements. CAN

messages and voltage measurements are time-correlated via 'tick count' held in a shared register.

The equipment used in the TCU voltage sensor characterization harness includes: (1) Power supply: Meanwell NES – 100, (2) Voltage Measurement: CompactRio with a National Instruments NI 9229 input module, and (3) CAN logger: CompactRio NI 9863 CAN 0 Logger.

Determining the voltage differential between measured and CAN logged voltages is described as follows: 1.) select a CAN voltage reading, 2.) match the supply voltage via tick count, and 3.) averaging the two closest supply voltage readings if no exact tick count match is available. Factoring in an inferred conversion lag (5 ms is suggested) uses the same process wherein CAN voltage readings have been adjusted accordingly.

The final step in the system characterization and analysis process is the simulation of the CAN voltage as generated by the TCU. In order to accomplish this, a highly-deterministic TCU simulation system is designed that listens for incoming OBD-II messages on the CAN bus and generates simulated CAN voltage messages matching those of the physical TCU as closely as possible. This design uses the Vector CANcaseXL logger instead of the CompactRio. The software features parameters accounting for the voltage underreporting, the CAN conversion lag, and the maximum response frequency encountered during the analysis of the physical device. The goal is to mimic TCU output, while providing adaptability that can be useful to investigate alternative scenarios.

Upon receiving an OBDII query, the system converts the latest supply voltage reading into a CAN message and sends it onto the CAN bus. The simulated TCU cannot use the same CAN ID as the physical TCU and instead uses CAN ID 0x7EA for disambiguation. The CAN messages sent out by the physical and simulated TCU are captured by the CAN logger and arranged for chronological computation of differential voltage.

2.1.5 Formal Verification Study

The investigation into formal verification techniques begins with an assessment of various formal models for hybrid systems. These are evaluated against expressiveness in modeling, potential for analysis, and tool support. A canonical vehicular CPS element is used as the object of formal analysis, and studied using the candidate system.

2.2 Passenger Car Event Data Recorder Research Methods

This study required the collection and interpretation of CAN data. In the following chapter the devices and methods used to record and interpret CAN bus traffic are explained and a brief overview of the study is given.

2.2.1 Methodology Overview

Fundamentally, the data stored on the Honda SRS module can come from one of two places: 1. the internal sensor circuits or 2. the messages existing on the CAN bus. The source for pre-crash information typically comes from the CAN messages and the SRS internal accelerometer provides the data source for the delta-V data. Therefore, if the CAN message data is known at the time of recording, then the data storage mechanism can be systematically studied by comparing the retrieved event data to the known data on the CAN bus. A graphical depiction of the basic methodology used for this paper is presented in Figure 4 as a flow chart.

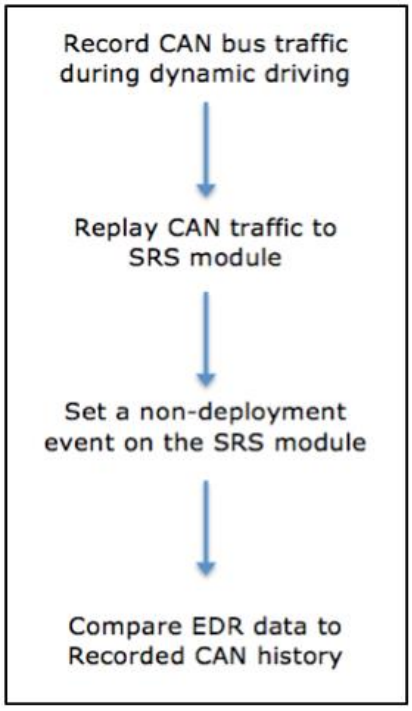


Figure 4: Methodology Overview

2.2.2 Driving Tests

Data was collected from the two test vehicles under a variety of driving conditions including steady state, maximum ABS braking, acceleration, Figure 8's, yaws, and normal driving. For the maximum ABS braking condition, multiple test runs were made from a highway speed (approximately 113 km/h or 70 mph) and multiple runs from a lower starting speed (80 km/h or 50mph). These data were archived so they could be used as sources to replay the data back to the SRS module in the lab.

Each car was instrumented with a Racelogic VBOX 3i and a Vector CANCaseXL logging device. The VBOX was programmed to transmit 100 Hz GPS-based vehicle speed on CAN ID 0x302 and time on CAN ID 0x301 over the vehicle's CAN bus according to the VBOX 3i User's Manual. To ensure that no data would be lost because of this transmission, CAN data was logged before the addition of the VBOX

and it was determined IDs 0x301 and 0x302 were not used by the Honda CAN network, making them available for VBOX data. An example of CAN data from a hard-brake test for the Honda Civic and Honda CR-V are shown in Figure 5 and Figure 6. It contains CAN data that show the engine speed in RPM, indicated vehicle speed, accelerator pedal position, and brake status. Additionally, it shows the VBOX speed that was transmitted onto the CAN network. This is fundamental to this research, because it enables all data to be synchronized by using the same data bus. While injecting the VBOX data onto the CAN, the busload hovered around 40%, thus no issues regarding normal CAN bus behavior and timing were suspected.

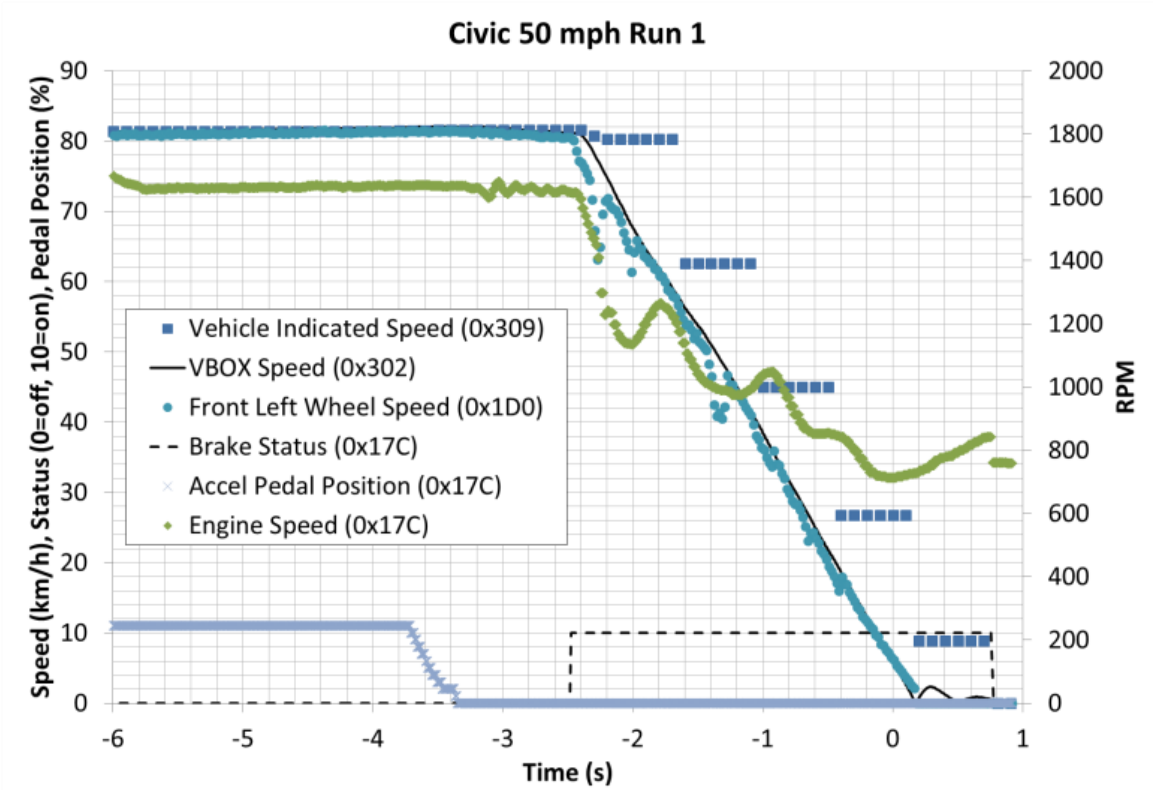


Figure 5: CAN data showing a hard brake on the Civic from 50 mph

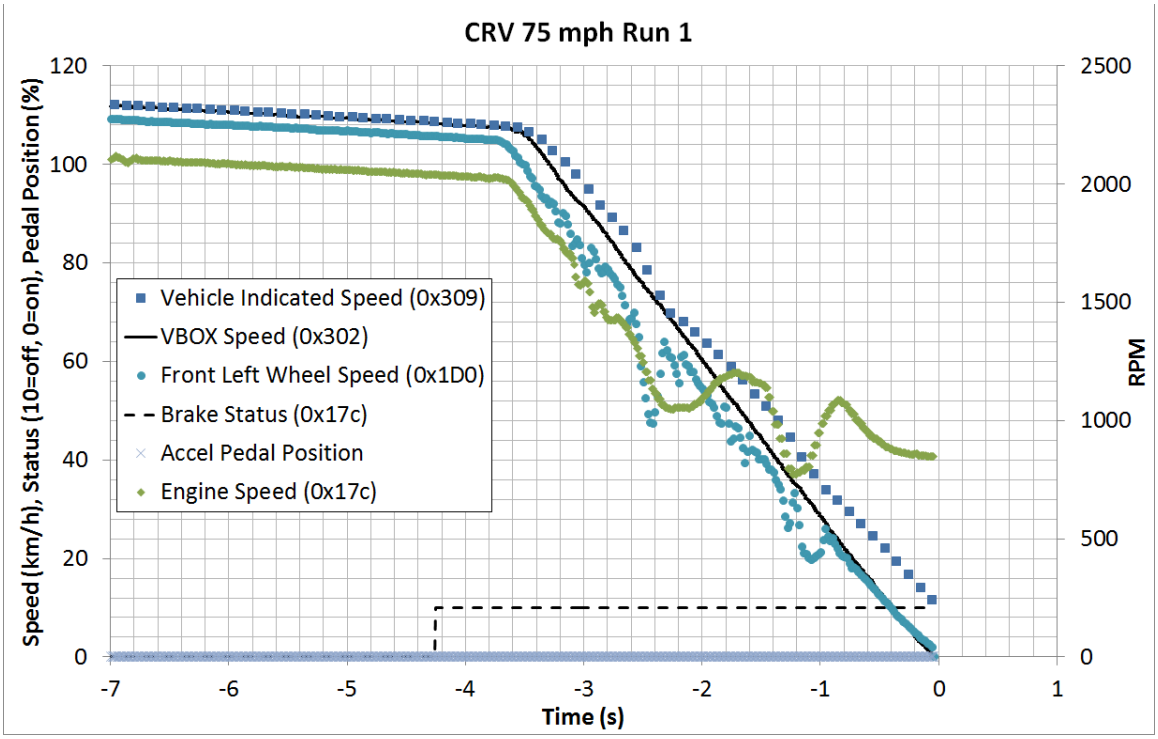


Figure 6: CAN data showing a hard-brake on the CR-V from 75.

All CAN data were logged to a Vector CANCaseXL Log device that recorded time, message spacing, ID, Data Length Code, Data entries, and Bus statistics. These binary files were converted to a text based log file and parsed to extract the data of interest. Because these files contained all the CAN data, the file sizes were large. Therefore, the runs used for replay were trimmed to nearly 7 seconds so the hard brake events were captured along with the steady-state speed section proceeding as shown in Figure 5 and Figure 6. Some of the non-interesting CAN messages were removed from the dataset since they had no meaning to the SRS module. These data were transformed and stored so they could be used repeatedly.

2.2.3 Interpretation of CAN Messages

CAN messages logged by the Vector CANCase XL Log were saved as a tab delimited files in the format shown in Figure 7. To interpret these logged files two things must be determined: message location (i.e. CAN ID and byte(s) comprising message) and bit resolution of data.

```
;Timestamp [ns]→Timediff[µs]——→Trigger Type——→Event Type→Data——→Data——→Data——→DataCRLF
0——→+1344368960000000µs→——→Realtime (83)——→Tue Aug 07 19:49:20 2012CRLF
0——→+0µs——→Single→Session Start (4484)——→ID:c0015b09CRLF
0——→+0µs——→On/Off→BUS INFO (4c2)→ch:1——→Baudrate:1441, Silent:0, Protocol:0, Pre-Trigger:0, Post-Trigger
0——→+0µs——→On/Off→BUS INFO (84c2)→ch:2——→Baudrate:1441, Silent:0, Protocol:0, Pre-Trigger:0, Post-Trigger
192297112000——→+192297112µs——→On/Off→CAN (497)——→RX ch:1→Dlc:7——→id:305→data:00 a8 00 00 00 00 3b CRLF
192297444000——→+332µs→On/Off→CAN (496)——→RX ch:1→Dlc:6——→id:156→data:ff d0 00 00 07 19 CRLF
192297907000——→+463µs→On/Off→CAN (498)——→RX ch:1→Dlc:8——→id:91——→data:80 00 08 05 fb 00 00 3c CRLF
192298155000——→+248µs→On/Off→CAN (498)——→RX ch:1→Dlc:8——→id:1a4→data:00 00 00 00 00 00 00 09 CRLF
192298397000——→+242µs→On/Off→CAN (498)——→RX ch:1→Dlc:8——→id:1aa→data:7f ff 00 00 00 00 66 03 CRLF
192298625000——→+228µs→On/Off→CAN (497)——→RX ch:1→Dlc:7——→id:1b0→data:00 0f 00 00 00 00 0d CRLF
192298850000——→+225µs→On/Off→CAN (498)——→RX ch:1→Dlc:8——→id:1d0→data:49 bc 93 91 25 32 4b 14 CRLF
192299100000——→+250µs→On/Off→CAN (498)——→RX ch:1→Dlc:8——→id:1ea→data:00 00 00 fa 00 00 00 06 CRLF
```

Figure 7: CAN Case XL Logger Exemplar File.

2.2.3.1 Determination of Message Location

To determine the message ID on the CAN network, human ability for pattern recognition was utilized. To consider an example of this process a VBOX GPS speed record is shown in Figure 8 for a hard-brake with speed plotted on the y-axis in km/h and time plotted on the x-axis in seconds. For now, we will consider only the shape of the curve.

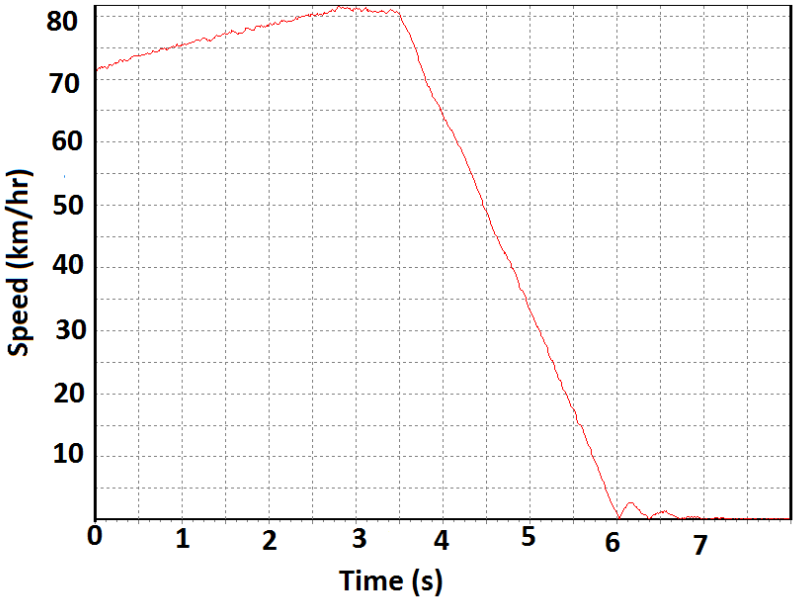


Figure 8: Example VBox GPS Speed Record.

For this CAN history, plots were created for each CAN ID systematically for single bytes and byte concatenations. The plots for ID 0x309 are shown below in Figure 9 and Figure 10. The shapes of CAN ID 0x309 bytes 4 and 5 in Figure 10 and the VBOX speed plot above match, indicating bytes ID 0x309 bytes 4 and 5 as a good candidate for the vehicle speed message. Other CAN histories were considered and this ID and byte combination was determined to be vehicle speed.

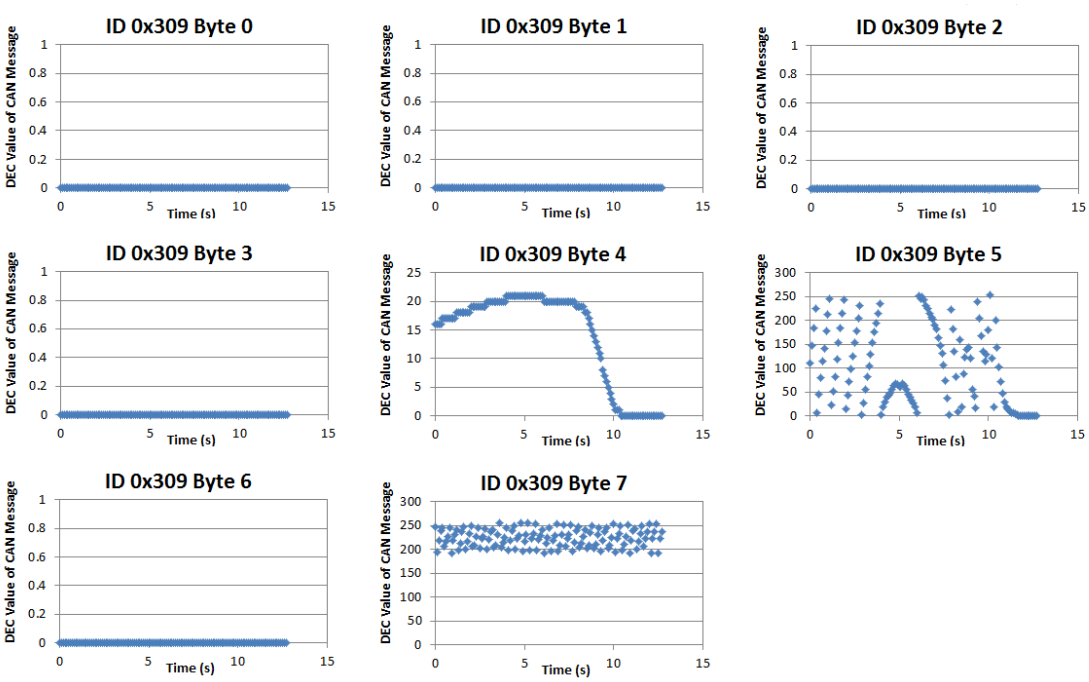


Figure 9: ID 0x309 Single Bytes plotted vs. time.

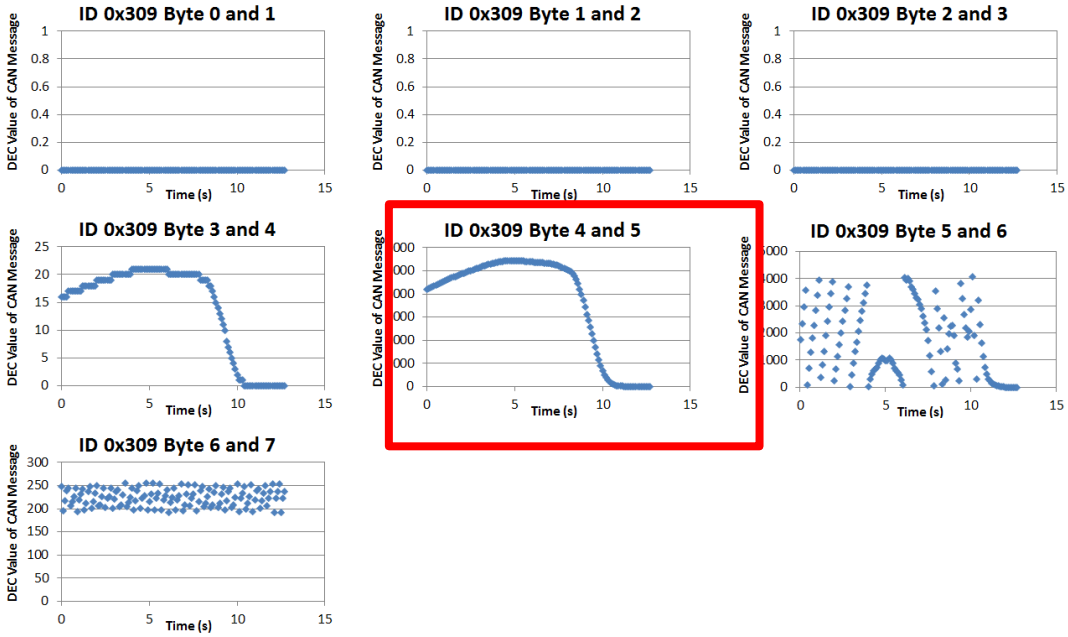


Figure 10: ID 0x309 concatenated bytes plotted with time.

To automate this plotting process a Python script was written to systematically graph CAN data vs. time. The plots are generated with the most significant byte first. For example, if byte 4 = 0x17 and byte 5 = 0xc4, the concatenated bytes would be 0x17c4. Since byte 4 appears first in the concatenated string the byte order follows the big-endian or Motorola format.

2.2.3.2 Determination of Bit Resolution

To determine the bit resolution of ID 0x309 bytes 4 and 5, the VBOX GPS based speed was divided by the decimal value of the CAN message providing a calibration factor between the CAN decimal value and speed in km/hr. This methodology was used to identify CAN messages that may serve as SRS sources and a summary of the results is given in Table 1Table 3. In this table the likely conversion rate is given in terms of the least significant bit (LSB). For example, take 0x17c4 as a speed message. This value converted to decimal is 6084. If we multiply the decimal value by 0.01 km/h, as given in Table 2.1, we have a resultant speed of 60.84 km/hr.

Table 3: Honda SRS Sources

Quantity	CAN ID	Byte(s)	Likely Conversion Method	CAN Refresh Rate (s)
Speed Vehicle Indicated	0x309	4 and 5	0.01 km/h per LSB	0.1
Accelerator Pedal Position	0x17c	0	0.5% per LSB	0.01
Engine RPM	0x17c	2 and 3	1 rpm per LSB	0.01
Service Brake	0x17c	Bit 0 of Byte 4	1 = On, 0 = Off	0.01
Steering Wheel Angle	0x156	0 and 1	-0.1 degree per LSB, signed integer	0.01

2.2.4 CAN Replay System Design

In this chapter the CAN replay system requirements and design will be explained. The CAN replay system consists of the mechanical testing apparatus, the electrical design which accompanies it, and the LabVIEW software implementation which controls it.

2.2.4.1 Mechanical Design Requirements

The SRS modules need to experience an acceleration that will enable the recording algorithm. According to the Data Limitations for Honda in Version 8.1 of the Bosch CDR report: “A non-deployment event is recorded if the change in longitudinal or lateral velocity equals or exceeds 8 km/h over a 150 ms timeframe…” The CDR report also specifies, “A deployment event is recorded if front airbag(s), side airbag(s), or side curtain airbag(s) are commanded to deploy. Deployment events are locked into memory and cannot be over-written.” The permanent writing of an event to a module renders that module useless for further study. Thus deployment events must be avoided or only one data point may be obtained per module. However, if non-deployment events are achieved, an SRS

module may be used for many tests. Since the achievement of non-deployment events are essential to this study, a detailed description of the methods used to ensure only non-deployment events would be generated are presented in the next sections.

2.2.4.2 Mechanical Design

The event generation apparatus consists of a linear sled and a pneumatic cylinder that was designed to provide a delta-V around 10 km/h in less than 150 ms. This deceleration occurs as the carrier with the SRS module reaches the end of the motion and comes to a stop. An external accelerometer was used to measure the accelerations and independently determine the delta-Vs. An annotated photograph of the test setup is shown in Figure 11, a schematic shown in Figure 12, and a skeleton drawing shown in Figure 13.

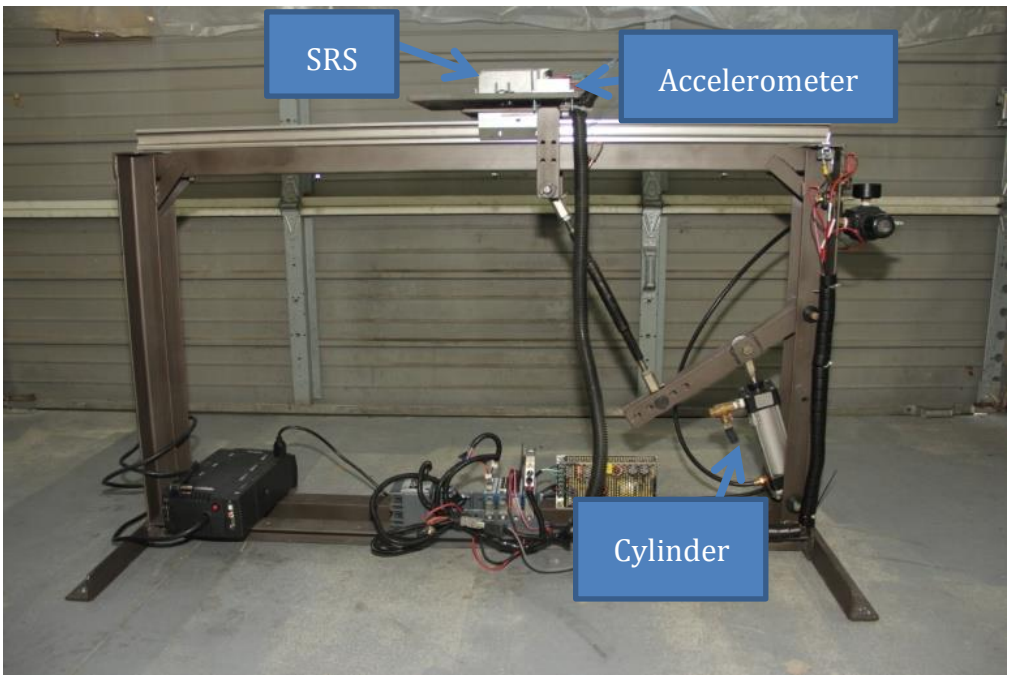


Figure 11: Mechanical test fixture.

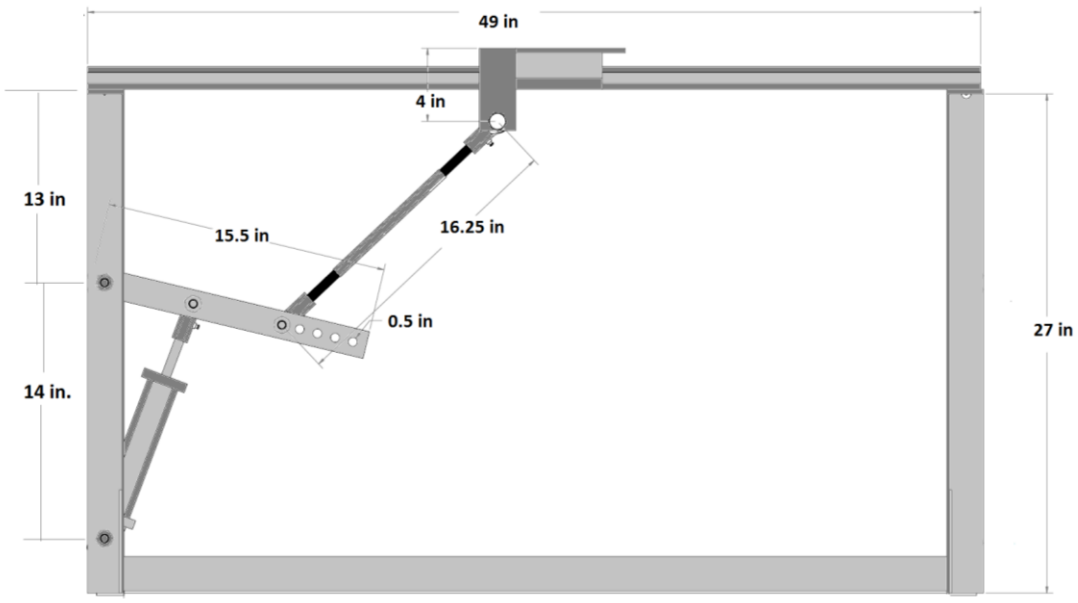


Figure 12: Non-deployment event generation skeleton drawing dimensions

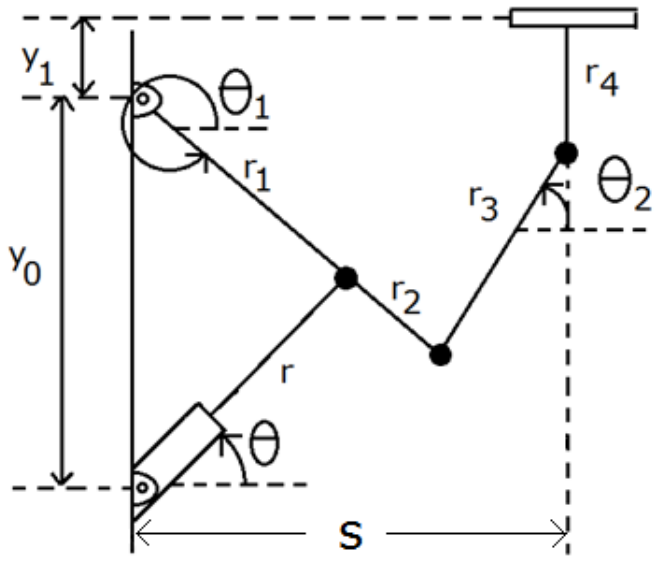


Figure 13: Non-deployment event generation skeleton drawing variables defined

Since a repeatable acceleration profile was necessary to achieve non-deployment events, a kinematic analysis of the apparatus was done. For this analysis loop closure equations were written to predict the acceleration magnitude and duration the EDR module would experience upon firing of the apparatus. Figure 13 provides the definition of the variables used in the kinematic equations.

Once the testing apparatus was constructed the acceleration pulse was measured using a Spectrum 15200B ±35 g accelerometer sampled at 4000 Hz for various solenoid pressures. Figure 14 shows an acceleration plot produced during a test. In this figure the acceleration pulse is plotted as the solid line, the delta-v value from 20 ms is plotted with a dotted line, and the delta-v from t_0 is plotted with the dashed line. The delta-v from 20 msec data takes the delta-v from a 20 ms window while the delta-v from t0 data plots the accumulative delta-v value. The delta-v values were calculated using the trapezoid rule using different starting and ending criteria. The t_0 value referenced by this figure is the time at which the change in longitudinal velocity equals or exceeds -0.8 km/h over a 20 ms timeframe. This trial produced a non-deployment event and the corresponding pressure (65 psi) was used for testing. Additional testing proved that the acceleration pulse was consistent and that non-deployment events were achieved.

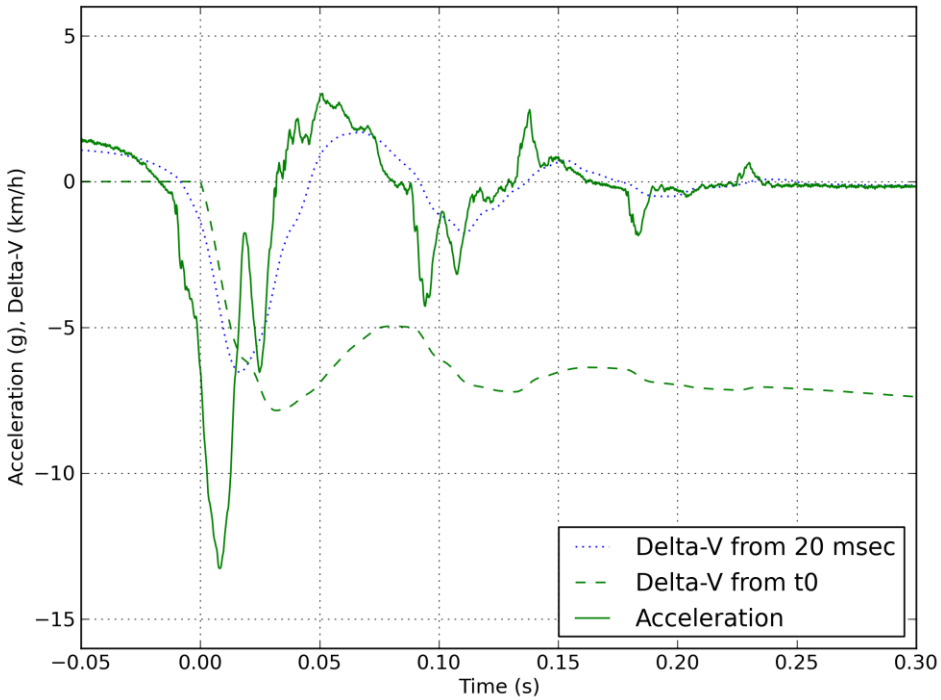


Figure 14: Non-deployment apparatus acceleration pulse.

2.2.4.3 Electrical Design

The actuator is a pneumatic cylinder with a stroke of approximately 10 cm. The air sent to the pneumatic cylinder was controlled using a pressure regulator in series with a solenoid valve. This solenoid valve is normally closed and is opened in one of two ways: 1.) a manual switch located near the rail or 2.) the output from the National Instruments NI 9478 digital output (DO) module. The DO module switch is controlled from the field programmable gate array (FPGA) program and was engaged based on a user specified time, which enabled an event to be generated during a specific section of the CAN history. Another switch was used as a start/stop

switch that began and ended the program. It ensured the cylinder was only actuated when the user was ready and the area was safe.

A Spectrum 15200B ±35 g accelerometer with a DC-400 Hz response was used to measure the acceleration the SRS unit experienced. The acceleration trace was time correlated to the CAN message transmission. This synchronization allows Algorithm Enable (t_0) to be established in the external CAN data sent to the SRS module.

The electronic communication schematic of the test apparatus is shown in Figure 16. The CAN network connects both CAN ports of the NI 9853 high speed CAN module, OBD II port, SRS module, and the female banana plug test points. The CAN information is transmitted from the CAN0 port of the high speed CAN module and recorded by multiple devices (SRS module, CAN1 port, and possibly the banana plug test points). The Honda SRS module requires an addition wire, the K-Line, to be connected for communications with the Bosch CDR tool. The SRS connector pin-out is shown in Table 3.1 and the SRS pin out is shown in Figure 15.

Table 4: 2012 Honda SRS Connector A and OBD-II pin out.

SRS Pin	Signal	OBD-II Pin
19	K-Line	7
20	CAN H	6
21	CAN L	14
36	GND 1	5
37	GND 2	5
38	VBAT 1	16
39	VBAT 2	16

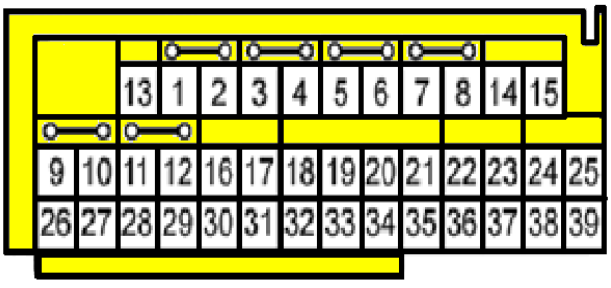


Figure 15: SRS Connector View Honda 2012 (Female)

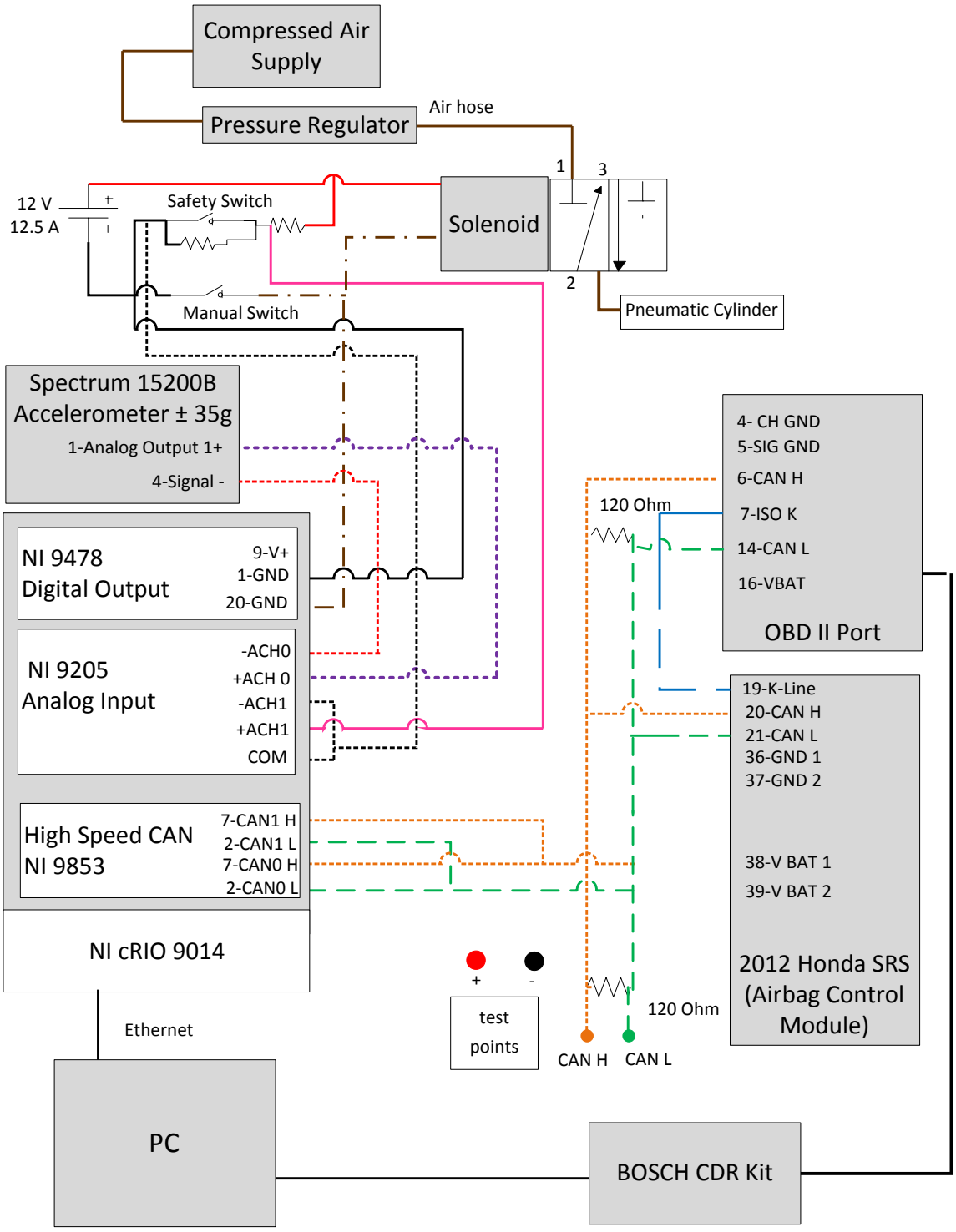


Figure 16: SRS module replay system schematic.

2.2.4.4 Software Design

Figure 17 shows the general overview for the software implementation used in this study. The implementation and devices will be described in the proceeding sections.

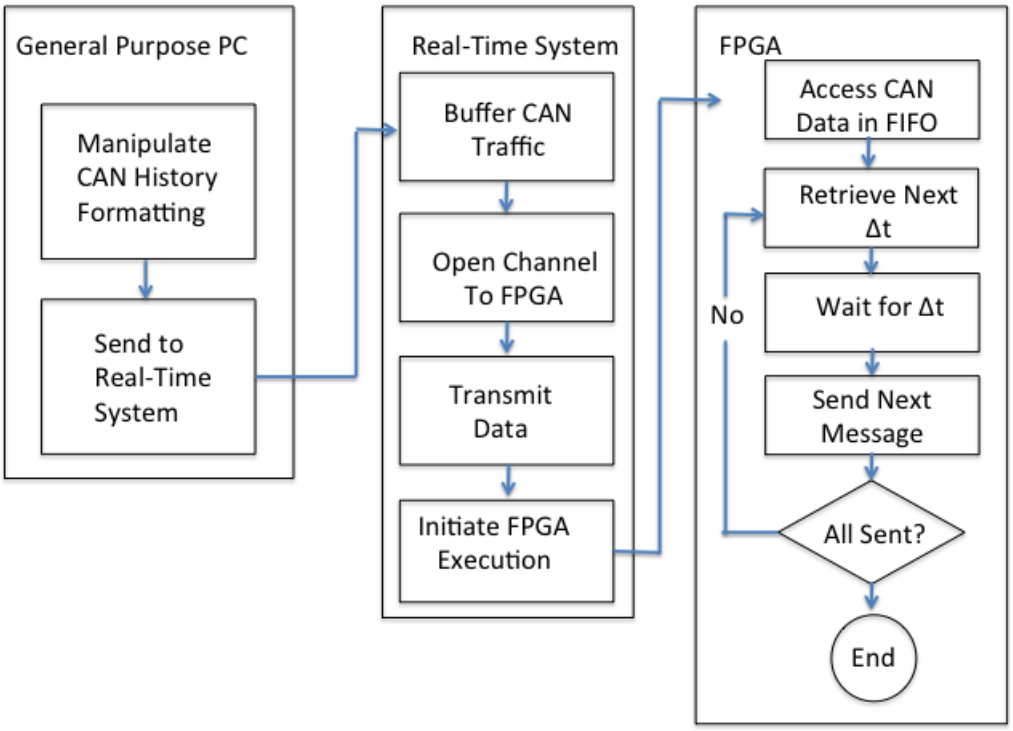


Figure 17: CAN replay methodology overview refinement.

The NI 9853 High Speed CAN module requires the transmitted messages be formatted into six unsigned 32-bit words: time high, time low, CAN ID, DLC, data 1, and data 2. For transmission, the time high and time low are set to zero. In order to comply with the standards required by this device a Python script was developed to parse and convert the CAN Case XL Logger file into an acceptable format. The program and process is described below.

To understand the Python script a screenshot of a raw vector file is shown in Figure 18 where arrows indicate tabs.

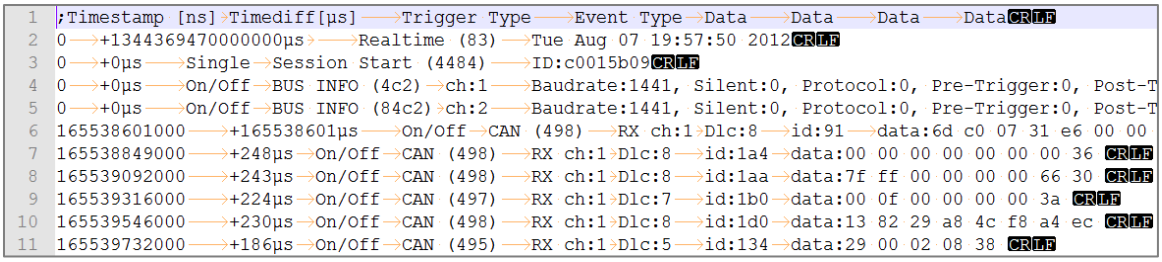


Figure 18: CAN CaseXL Logger Format

The CAN history was read into the Python program and each row was subsequently split into separate entries using the `.split('\t')` function, providing an array of data. Data was then appended to a one dimensional array according to its value. For example, `entries[6]`, which corresponds to the CAN ID, were appended to the "IDs" array. However, `entries[6]` for line 7 of Figure 18 contains "`id:1a4`". To report only the desired data, "`1a4`", `entries[6][3:]` were appended to the IDs array. A similar method was used for all data of interest in the CAN file.

The first data transformation necessary when using a CAN logger, like the Vector CANCaseXL, is to compute the time differential Δt between sequential messages as recorded by the logging hardware. This was achieved by simply subtracting the timestamps of two sequential CAN messages that are replayed. Furthermore, logging devices often captures data that was not conveyed via CAN messages and that can be discarded since it is logger specific and has no relevance for the temporal relationships. For example, the Vector CANCaseXL is capable of recording information concerning bus load. Since this information is not native to the Honda CAN network it can be stripped from the Vector log file to attain the Honda CAN history without altering the temporal relationship of the recorded CAN messages. The removal of superfluous data was achieved using lines 42-55 of the python script shown in Figure 19. If the data in `entries[6]` is not a CAN ID that row is skipped and not appended to the IDs array. Furthermore, only certain CAN IDs are needed for replay to the SRS module, thus a blacklist was created. If the CAN ID is blacklisted, it is skipped and not appended to the IDs array.

```
42      for line in lines[6:]:CRLF
43          entries=line.split('\t')CRLF
44          #print(entries) #(only for debugging)CRLF
45          try:CRLF
46              #build a list of IDsCRLF
47              if entries[6][2]!='x':CRLF
48                  identifier=entries[6][3:]CRLF
49                  if identifier in blacklist:CRLF
50                      continueCRLF
51                  else:CRLF
52                      IDs.append(identifier)CRLF
53              else:CRLF
54                  print('idx found... skipping')CRLF
55                  continueCRLF
```

Figure 19: Removal of logger specific data from CAN history file.

To ease data replay, all CAN messages can be stored in a flat file. The flat file contains rows of data in which the first column comprises the integer message spacing in microseconds, the next column is the decimal representation of the CAN ID, column 3 is the Data Length Code, and columns 4 and 5 are the CAN data as represented in decimal form using 32 bit words. In other words, column 4 contained the first 4 bytes of the CAN message and column 5 contained the last 4 bytes of the CAN message. If the data length code was less than 8, then the missing bytes were filled in with zeros and converted. The creation of the described flat file was achieved in lines 88-122 shown in Figure 20. In line 88 of this code the new file is opened and in line 121 data is written to the new file. To format our data into two U32 words (data 1 and data 2), lines 90-118 use byte shifting methods. For example, in line 103 the "`data1`" (i.e. bytes 0-3) are defined by shifting the 64 bit (8 byte) message, "`bigData`" 32 bits. This shift allows the less significant 32 bits to be stored.

```
    newFile=open(f[0:-4]+'_ForLabVIEW_CRV_brute_force.csv','w')
    for delay,ID,DLC,hexMessage in zip(delays,IDs,DLCs,hexpayloads):
        #print(hexMessage)
        data=''
        for h in hexMessage:
            #print h
            data+=h #Data[0] is the MSB
        #print(data)
        #print len(data)/2
        bigData=int(data,16)
        #print bigData
        if len(data)/2 == 8:
            data2=ctypes.c_uint32(bigData).value
            #print(data2)
            #print(hex(data2))
            data1=bigData>>32
        if len(data)/2 == 7:
            data2hex=data[-6:]+'00'
            data2=int(data2hex,16)
            data1hex=data[:8]
            data1=int(data1hex,16)
        if len(data)/2 == 6:
            data2hex=data[-4:]+'0000'
            data2=int(data2hex,16)
            data1hex=data[:8]
            data1=int(data1hex,16)
        if len(data)/2 == 5:
            data2hex=data[-2:]+'000000'
            data2=int(data2hex,16)
            data1hex=data[:8]
            data1=int(data1hex,16)
        #print(data1)
        #print(hex(data1))
        newFile.write('%i,%i,%i,%i,%i\n' %(delay,int(ID,16),int(DLC),data1,data2))
    newFile.close()
```

Figure 20: Python implementation of creating a flat file after data conversion.

2.2.4.5 Design Requirements for CAN Transmission

The challenge of the methodology is minimizing external effects on the replay algorithm that could interfere with the temporal relationships contained in the recorded CAN traffic. A simple replay implementation would be to use any general purpose programming language to implement and execute a replay algorithm on a general purpose PC. The problem with this approach is that most operating systems used on general-purpose computers do not guarantee the timeliness when a certain function within a program is executed. Furthermore, the relative execution timing might vary depending on other processes currently using the system. Thus, there is no guarantee that the replay algorithm is ready to send a CAN message at a specific point in time, since the process might not currently have access to the CPU. Additionally, repeating the replay algorithm is likely to generate significant variation in the message timing since CPU load varies over time.

To minimize distortions in timing between messages the steps comprising the overall CAN replay system were grouped according to the time sensitivity and

implemented on platforms that best meet their respective needs. A combination of a general purpose PC (Windows 7 running LabVIEW 2011) and a real-time system with field programmable gate array (FPGA) (National Instruments CompactRIO) was suitable to achieve the goal of minimized temporal distortion.

2.2.4.6 Real-Time Operation System

A real-time operating system is a special-purpose operating system that imposes rigid time requirements on process execution. Among real-time operating systems two subcategories are frequently distinguished. A hard real-time system guarantees that all delays within the system are bounded via an upper and a lower execution time that must be met at all times. To achieve this, the set of available functions is limited and algorithms using such systems must be designed to achieve their goals with the available functions. A soft real-time system does have upper and lower bounds for all functions but it assigns and manages varying levels of task priorities.

In this study, a hard real-time system was used to interface the messages stored in the flat file and the FPGA. This was done because the FPGA was not able to store the entire CAN file used for replay. Thus the real-time system was used to transmit the CAN file into a FIFO (which acts as essentially a buffer) passing the flat file to the FPGA allowing it to be replayed. This forwarding process is done without imposing a strain on the timing between CAN messages.

2.2.4.7 Field Programmable Gate Array

A field programmable gate array (FPGA) is an integrated circuit that can be configured via an appropriate hardware description language. It combines hardware-typical speed, determinism, and reliability with some of the flexibility of general purpose programming languages. An FPGA allow several programs to execute truly parallel without competition for shared resources and offer nanosecond response times for input-to-output processing. Of course, offering such a feature set comes at the expense of the complex FPGA operations, meaning that not all algorithms are suitable for FPGA implementation and compile times are longer. Due to the need for accurate timing and synchronization, the FPGA was used as the core technology of the system.

2.2.5 Software Implementation

The software used to run the system was written in LabVIEW using the following development targets: (1) Python programing on a PC, (2) real-time system, and (3) a FPGA. The LabVIEW programming environment eases development and testing across targets.

The tasks executed using Python are related to data processing and storage. First, the log files created by the CAN message logger are processed to extract the relevant message time stamps and the data conveyed via the CAN messages. Everything else is ignored and only the relevant data is designated for storage. The CAN replay files were prepared by placing them into a comma separated values table and uploaded to the real-time controller using FTP (file transfer protocol). After the test was

finished, Python was used to post-process the data and produce values with engineering units.

The main function of the real-time system included in the CompactRIO platform is to provide the data prepared using Python to the FPGA. As described previously, the FPGA does not have the necessary storage space for the CAN history files to be stored. To overcome this limitation, the real-time system writes the CAN history of interest to a FIFO (first in first out) which is shared by the FPGA. The bock diagram of the real-time VI written for this project is shown in Figure 21 and Figure 22. In the lower left section of Figure 21, we see the case structure window in which the CAN history used for replay is selected. Using the FTP, we are able to upload multiple CAN histories on the real-time system. The front panel of this VI has a selection window in which a list of available CAN histories are given. Once the CAN history has been selected, the length of the data file is determined (both time and number of messages) and used to configure the CANDataFIFO. The data is then written to the FIFO in the second window of the flat sequence of Figure 21. The solenoid delay value, shown in the first flat sequence window of Figure 21 is specified by the user in the front panel of this VI. The solenoid delay allows the user to determine when the non-deployment event will be created in the CAN history. Once the sum of the delays (time between synchronous messages) reaches the value of the solenoid delay the NI 9478 DO module switches, providing the necessary voltage to actuate the cylinder (rig schematic provided in Figure 3.7). The real-time VI is also used to write the output files of the experiment. As shown in Figure 22, the real-time VI opens two new files: Accel.bin and TransmittedCAN.bin. These files contain the acceleration record of the external accelerometer and the transmitted CAN history. These files are rewritten every experiment and must be taken from the real-time system following each experiment using WinZip. The values of these files are generated in the FPGA and are transmitted to the real-time system using FIFOs.

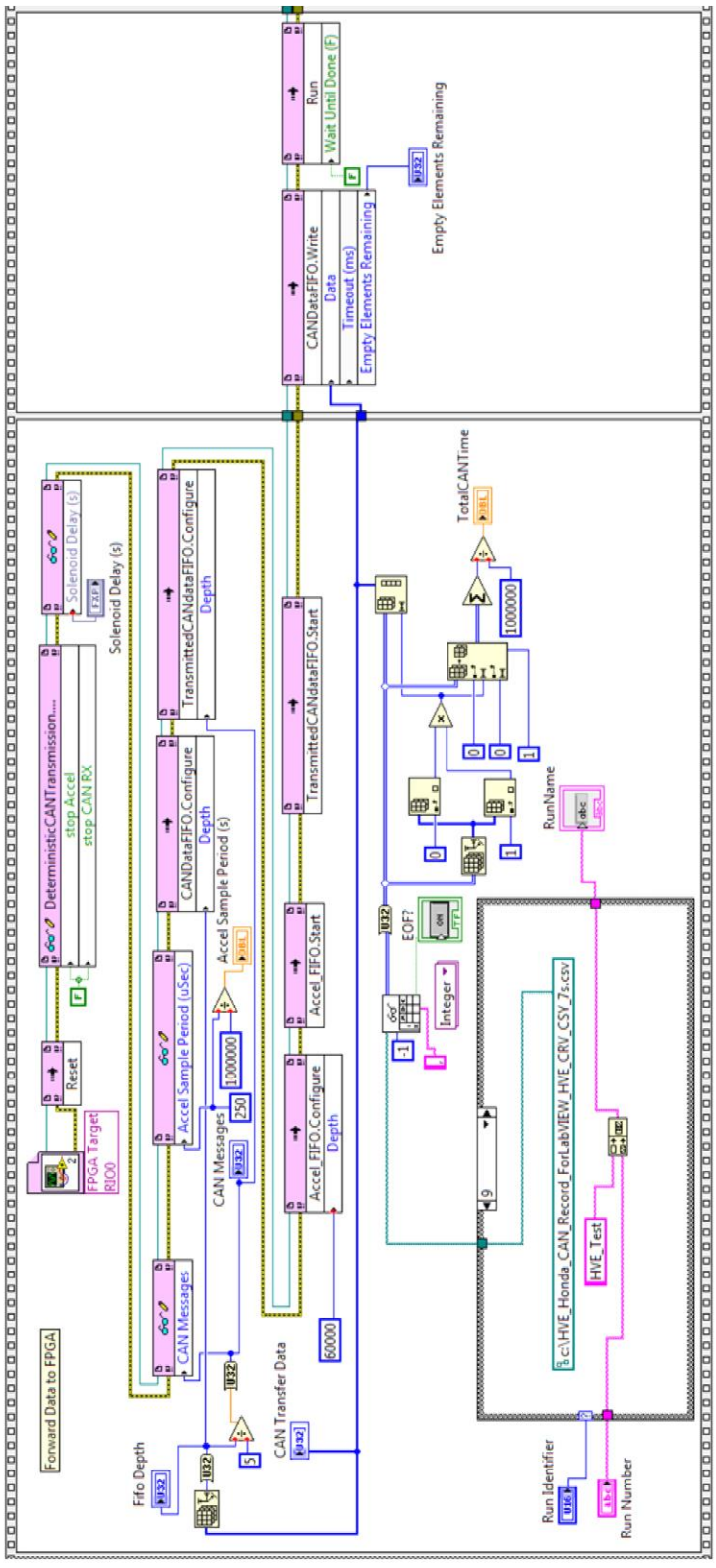


Figure 21: LabView Real-Time VI: FTP Reading and FIFO Configuration

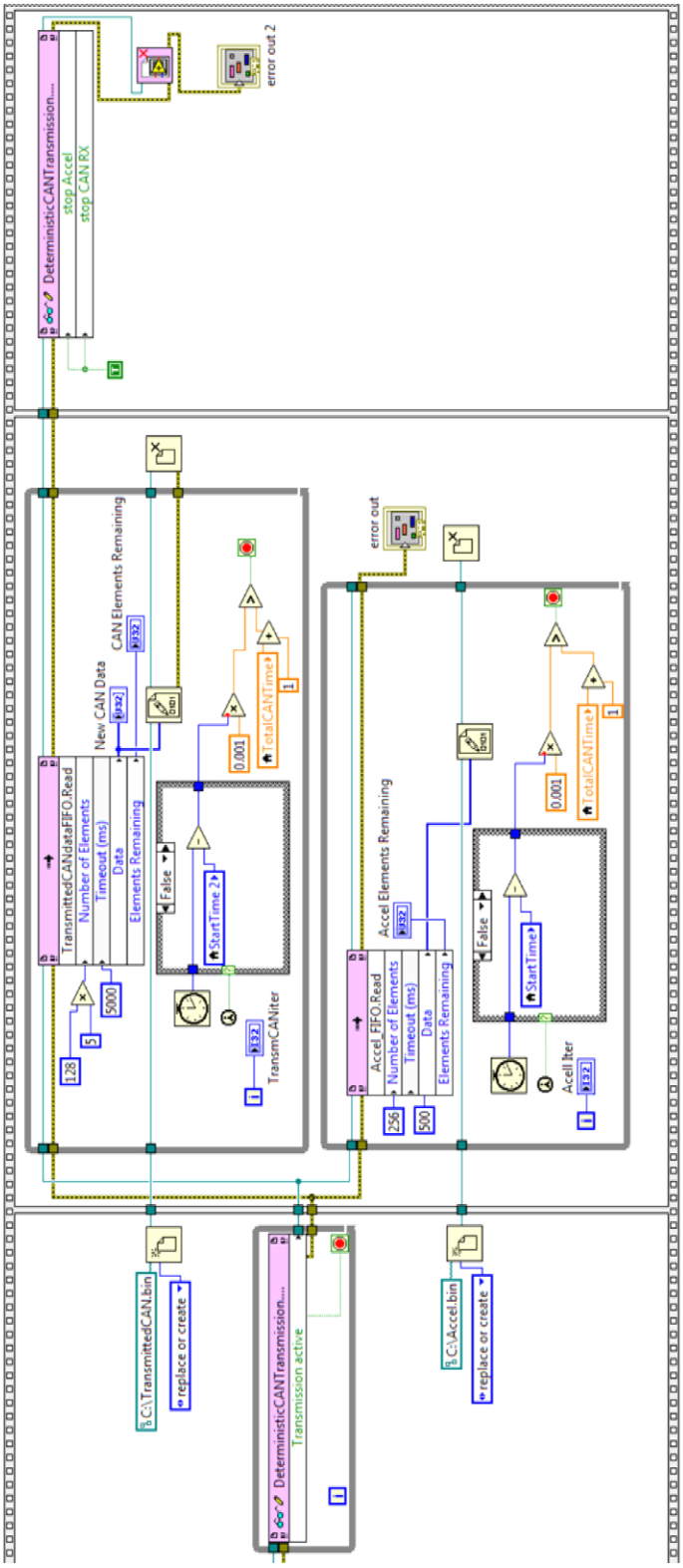


Figure 22: LabVIEW Real-Time VI - writing binary data files.

The LabVIEW program implemented for the FPGA is the core technology that enables real-time processing. The block diagram for LabVIEW is shown in Figure 23 and comprises three distinct blocks. First the flat sequence on the top is used to control the transmission of CAN messages from the FIFO established by the real-time VI. The For Loop in the center of the sequence is timed to cycle to the nearest microsecond according the delay measured between messages. These delays are summed and the solenoid is fired once a user inputted delay has passed.

The second block is a recording loop for the CAN data that were transmitted. This enables verification as to what the SRS module actually saw during the test. Furthermore, it produces a timestamp that can tie to the accelerometer recording function, synchronizing the CAN and accelerometer records. The loop executes aperiodically according to the received CAN message and times are attributed from the internal clock of the CAN module.

The third block, shown in the lower right of Figure 23 contains the acceleration sampling. This is a timed loop that executes at a user specified value (every 0.00025 seconds or 4000Hz in this case). The raw accelerometer data is converted to microvolts, combined with a timestamp from the CAN module, and sent to the real-time controller as a signed integer through a FIFO. The LabVIEW implementation on the FPGA enables the determinism in the timing needed to accomplish this research.

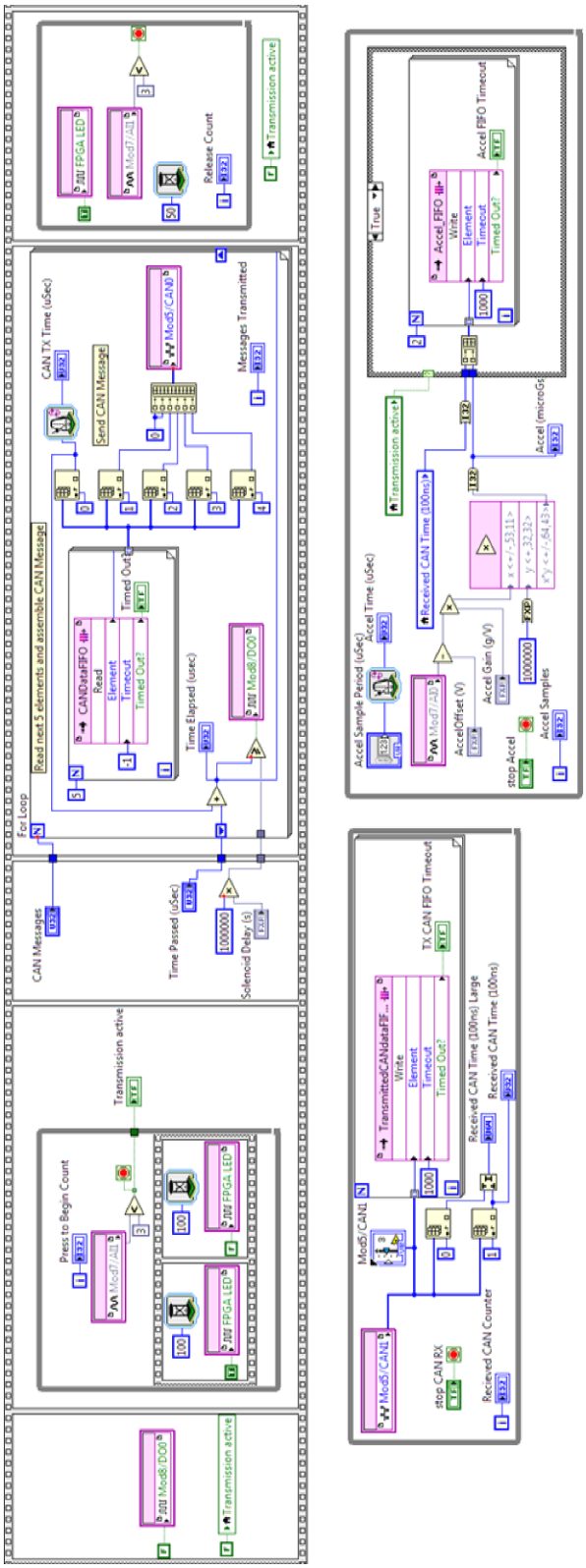


Figure 23: LabVIEW program implemented on the FPGA.

2.2.6 CAN Replay Experiments

A detailed overview of the experimental process is provided as a flow-chart in Figure 24

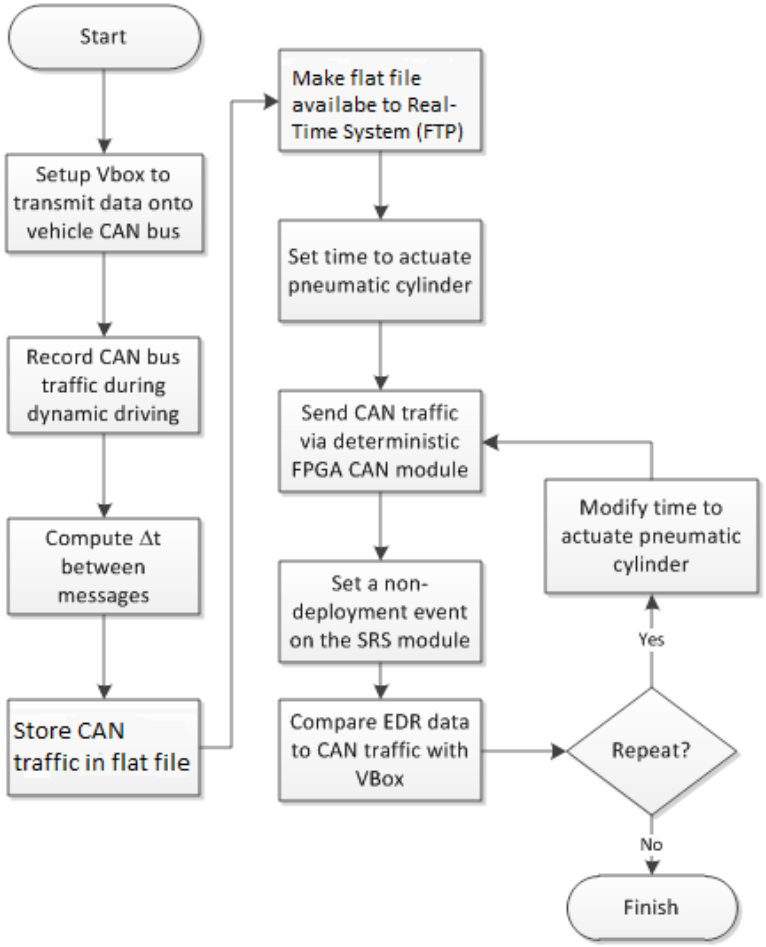


Figure 24: Details of the experimental process.

The test setup and software implementation described in this and the previous sections enables experiments to study the timing and accuracy of EDR data. The first set of experiments is to assess the accuracy of the CAN data compared to external references. To do this, the CAN messages were cataloged and characterized so values shown in the CDR report can be attributed to the correct CAN messages. Once some CAN messages were known, the CAN data can be compared to external references to gain a sense of the CAN data accuracy. Finally, the timing and data storage algorithms can be evaluated through reading SRS module data with the Bosch CDR kit after repeatedly setting non-deployment events for the same set of CAN message traffic.

Since determinism is paramount to this study two methods were used to verify the timing engines were true to the original data. First, the loop timer in the FPGA was

updated for each message based on the delay. Therefore, if all delays are summed, then the total run time is calculated. The predicted runtime can be compared to the actual runtime to get a sense for the determinism of the replay.

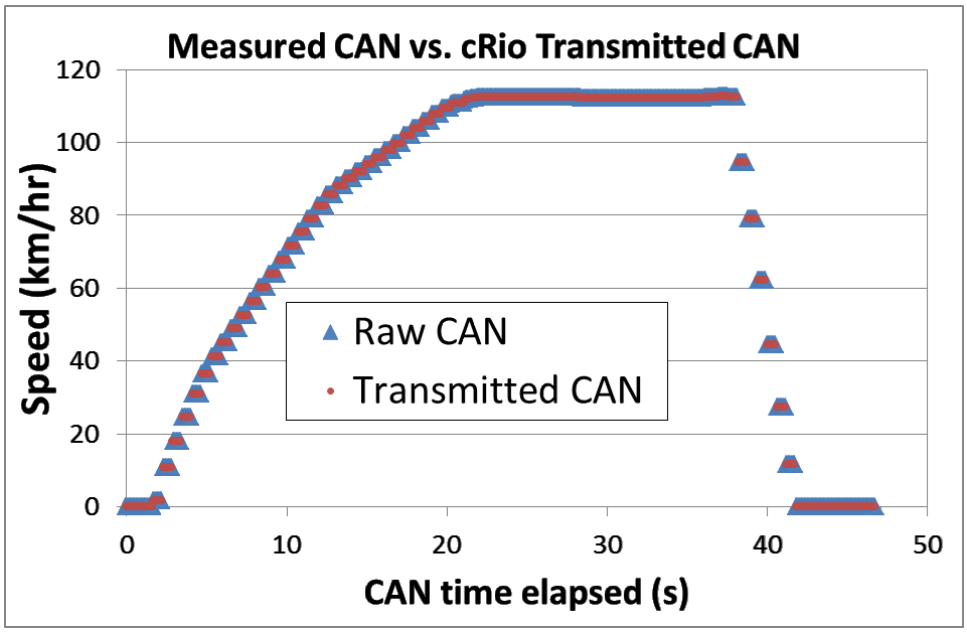


Figure 25: Chart of an example run from a 2012 Honda Civic showing a vehicle speed trace from CAN messages.

For example, the raw CAN messages in Figure 25 were obtained using a Vector CANCaseXL Log attached to the Honda Civic. The time between the first message with ID 0x309 and the last message with ID 0x309 was reported to be 46.708596 seconds. The replayed CAN messages were obtained from the CAN1 port of the National Instruments NI9853 that was used to record the messages sent during a test run on the non-deployment apparatus. The time separation of the first and last 0x309 message was 46.708202, which is a difference of 394 microseconds over this span. This suggests an average error rate on the replayed CAN message timing of 0.00084%.

Second, for each run the timing was verified by comparing the timestamp produced by the VBOX during the driving test to the recorded CAN from the replay. The VBOX 3i message encoding the time was also transmitted during the tests using ID 0x301 bytes 2, 3, and 4. This 24-bit integer represents the number of 10 millisecond intervals since midnight UTC according to the VBOX 3i user manual. If this time value was replayed and recorded while setting a non-deployment event with an accurate time base, then a graph of the VBOX time divided by 100 with respect to time in seconds would have a slope of unity, meaning exact time correlation. A slope of less than unity would indicate a delay in the replayed CAN messages and a slope of greater that unity means the replayed CAN messages are seen by the SRS module faster than the original messages in the vehicle. The standard deviation of the residuals of the line fit gives a sense of the jitter in the timing engines of the VBOX

and CAN message replay hardware of the FPGA and NI9853 CAN module. The slope of the VBOX time signal and the replayed timestamp are checked for unity for each run. An example of this check is shown in Figure 26. Based on these verification checks, the CAN replay system is representative of the actual CAN data transmitted in the vehicle.

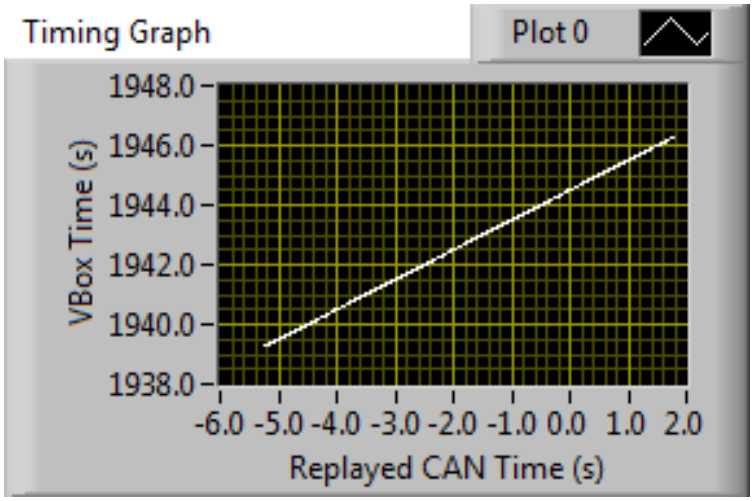


Figure 26: Timing verification graph that shows a slope of 1.00018.

2.3 Methods for Assessing Heavy Vehicle Event Data Recorders

Prior to developing and demonstrating the methodology, researchers outlined [37] forensic soundness issues with current ECM data extraction techniques, demonstrating avenues by which data may be intentionally altered or unintentionally corrupted. The main requirements of a forensically sound ECM data extraction technique are: data integrity, confidence in meaning of data, error detection and mitigation, and transparency and trust. Our solution, we believe, meets those requirements.

2.3.1 Requirements of a Solution

Regardless of the meaning of the digital data, it is necessary to present data in its final form in such a way that is transparent of its handling to establish trustworthiness.

According to the transparency principle of forensic soundness, actions taken by an investigator should be available for later examination. Additionally, any error conditions encountered by the software should be recorded so that the legal weight of the evidence may be accurately considered. Audit trails are log files generated by forensic software to meet these requirements, and should at least be an option in any forensic solution. Any solution must be able to trust that the ECM is reporting the data faithfully, be able to interpret the various protocols and message types (used in a vehicle network), provide an authentication mechanism that can be used to preserve data integrity, and preserve evidence even in cases where some data elements are not permanent such as system clock values.

2.3.2 Proposed solution for forensically sound extraction of ECM data

The proposed solution is based on proven techniques used in the world of computer forensics. The idea is relatively simple: make a low-level copy of the evidence drive is made (this copy is known as the ``disk image'' and then perform all forensic analysis on this disk image instead of the original disk (to minimize changes to the original source evidence). A vehicle network is different than a typical computer system and this idea or concept had to be extended to truck ECMs. This is done by making sure the information would be extracted exactly one time, and then replayed for further analysis. The replay traffic information would be stored securely to prevent malicious tampering or accidental alteration.

Development of the forensic replay software requires the following steps: (i) record the original information extraction process, (ii) decipher any encryption or obfuscation mechanisms obscuring ECM data, (iii) store the evidence in a secure manner and (iv) respond to information requests identically to the ECM.

There are a couple of differences between the proposed extraction replay method and a forensic disk extraction. One is the difference in the amount of data that is extracted. Rather than saving the entire contents of a disk, a replay of a software extraction only extracts the information normally accessed during that data extraction. It is possible that some relevant information exists on the ECM that is not extracted. The other difference in that while protocols for accessing hard disks are standardized, ECM data is often accessed with proprietary protocols, that vary from manufacturer to manufacturer, over standard networks. Therefore, for each individual manufacturer that will be supported, an understanding of the manufacturer's proprietary protocol extensions is required.

2.3.2.1 Data extraction and recording

It was determined that the best way to define the extraction process was to record the messages sent by the maintenance software to the ECM. Recording these messages could take place at two information boundaries, the network or the diagnostic link connector driver.

Communication between the ECM and the maintenance software may be recorded at the network level, but this method has some drawbacks: the need for a physical connection, specialized network logging equipment and ability to interpret various protocol (it is not uncommon for ECM communications to take place over both J1708 and J1939), and the loss of meaning due to use of RP1210 (by maintenance software) if only network-level observation is used.

The proposed solution uses an alternate method of recording information extraction based on recording calls made to the RP1210 drivers by the maintenance software. This has the advantage of not requiring additional hardware to record network traffic, and it abstracts away the details of transport-layer operations of J1939.

Recording calls made to these RP1210 drivers was accomplished using a technique called API hooking [40]. A custom lightweight debugger was written using the PyDBG tool. Upon attaching the debugger to the software in question, the debugger searches process memory for any loaded RP1210 drivers. Upon discovering a loaded RP1210 driver module, it places breakpoints on the following RP1210 function addresses: RP1210_ReadMessage, RP1210_SendMessage, RP1210_ClientConnect, RP1210_SendCommand and RP1210_ClientDisconnect.

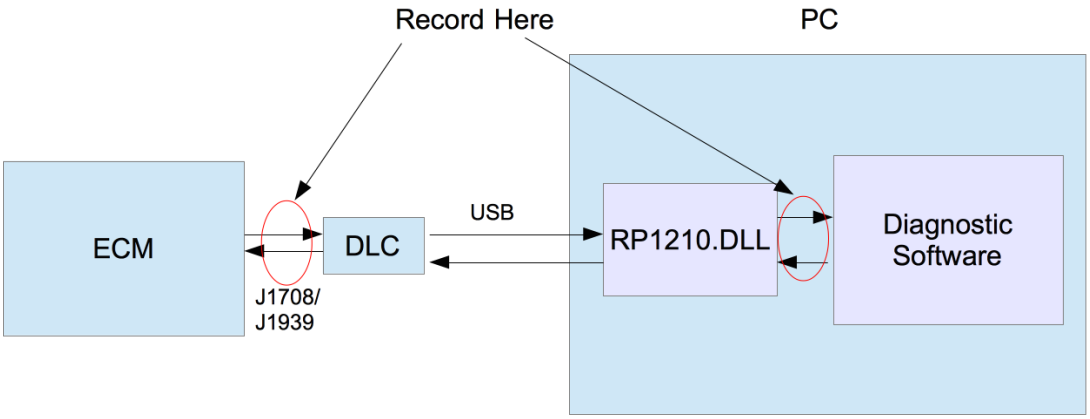


Figure 27: Data recording for proposed solution

When execution hits one of these breakpoints, the debugger reads all arguments passed to the function when it was called, and places a breakpoint on the function's return address so that return values can be read as well. Using this approach, all messages sent to and received from the ECM are recorded by the debugging software (Figure 27: Data recording for proposed solution).

This approach has an advantage that it doesn't require any additional hardware. Since the price of commercial vehicle network loggers can reach several hundred to several thousand dollars, cost-effectiveness should not be overlooked. We consider this an important contribution of this work. It also shows exactly which messages the application receives, which aids in determining which messages are important.

2.3.2.2 Replaying an Extraction

In order to ensure that the method of evidence extraction was as general as possible, evidence extraction was implemented by simply replaying requests recorded during an actual information extraction. In the case of encrypted requests, where the same request may be encrypted differently depending upon details of individual sessions, a transformation function to decrypt the message and store it in a plaintext format needs to be specified.

After each request is replayed, responses to that request are recorded. The extraction data are stored as a list of key-value pairs, where the key is the request (transformed to a plaintext format, if applicable) and the value is a list of all messages that the ECM sends in response to that request. This list is then serialized

into a format that can be stored in a file on disk; this file is the logical equivalent of a hard disk image.

As it has been observed that requests sent to the ECM may alter the state of the ECM, the replay mechanism must be designed so that this is taken into account. The stored replay data are treated as a circular queue, with a current index maintained during the extraction. Upon receipt of a message, the index is advanced until a matching response is found. If responses to requests depend on earlier messages, receipt of the earlier messages will advance the index to the expected response.

Just as a transformation function may need to be specified for extraction of encrypted information, such a transformation function may also need to be specified during replay. A comparison function needs to be specified, as the message format may preclude the use of straight comparison of request data.

The solution arrived at uses two separate threads of control, one handling J1708/1587 communications and the other handling J1939 communications. Each has its own protocol-specific response queue, though the two queues are both stored in the same file for evidence storage and encryption.

2.3.2.3 Hardware

The replay mechanism hardware is built around a BeagleBone, a commercially available ARM-based miniature computer produced by Texas Instruments [41]. The most recent iteration of the design, the BeagleBone Black, retails for roughly $55 and has a 1GHz ARM processor, 512MB of RAM, and 4GB of onboard flash storage. Weighing just a few ounces, it meets cost-effectiveness and portability requirements (Figure 28: BeagleBone platform).

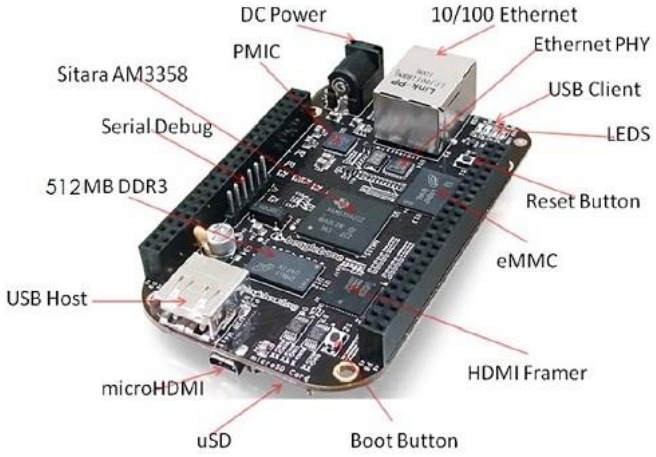


Figure 28: BeagleBone platform from the BeagleBone literature

One of the main reasons for adopting the BeagleBone platform is the capability to add functionality using expansion boards, known as Capes. Commercially available capes include a RS-485 cape and a CAN cape, supporting the physical layers of J1708/J1587 and J1939 with little to no modification.

As the currently available commercial options for RS-485 did not allow for communicating both over J1708 and J1939 networks, a custom hardware solution was required. A custom cape was designed for heavy vehicle communications based on J1708.

The J1939 network interface required CAN transceiver hardware. The BeagleBone includes CAN hardware on the board and driver support for CAN was already well-documented in the BeagleBone. All we had to do was implement the J1939 functionality on top of it. An existing implementation of J1939 for the Linux kernel was found and compiled into the BeagleBone's kernel as a module. As the remainder of the program was implemented in the Python programming language, the Python socket module was also patched to work with J1939. Also, J1708 software drivers for Linux were nonexistent, so new drivers had to be written.

2.3.2.4 Cryptographic Protection

In order to protect extracted data from alteration (or that would make it possible to detect any attempt to alter the data), a cryptography-based system was designed. The protection system was designed with the following requirements in mind: data must be protected from alteration, data must be protected despite the fact that all computation takes place on a device that is solely within the control of an unknown person and others' data must be secure even if a single device is compromised.

In traditional computer forensic investigations, a disk image is protected by performing a cryptographic hash on it. Later on, the image is hashed again and the two hashes are compared to confirm that the image has not changed.

In the case of ECM data records, however, the use case is somewhat different. The data are frequently extracted in remote locations where it is not feasible to have all parties to the case present. Therefore, the individual extracting the information has total access to the information being extracted, likely for a significant length of time. In this case, hashing alone may not offer the required protection as the data may just be altered and the hash recomputed. While this is also a risk in hard drive extractions, the possibility of having all parties present mitigates that somewhat.

Rather than attempting to protect the data from alteration, which is practically impossible with the device in the physical control of a potentially malicious actor, the solution is to strongly encrypt the data instead. If the data are strongly encrypted, while altering the data may be possible, **meaningfully** altering the data is practically impossible. By ensuring that an attacker will gain nothing by altering the data, the data are effectively prevented from being altered.

We use the following encryption algorithm was developed to perform this task:

1. A nonce, to be used as a symmetric key, is randomly generated.
2. The nonce is used to encrypt the data.
3. A public key, stored on the device, is used to encrypt the key.
4. The encrypted key is stored alongside the encrypted disk image.
5. Later, the RSA private key, stored with a trusted third party, is used to decrypt the symmetric key, which is then used to decrypt the data.

This is an example of a hybrid cryptosystem as described in \cite{cramer2004}. Cramer and Shoup prove that a hybrid cryptosystem of this type is secure so long as the underlying algorithms are secure, and the padding scheme used for encrypting the key is secure.

The symmetric algorithm chosen to protect the data is AES-128, as the AES algorithm is the industry standard symmetric encryption algorithm, and it is currently believed to be secure. The 128-bit key length was chosen because of breaks discovered in the 256-bit key length [42].

The cryptographic hash function chosen is SHA-256. While a longer hash value may yield better security, a longer hash may also make it more difficult to write down a hash value for an investigator in the field. SHA-2 was chosen over SHA-1 because of the widely-published attacks on SHA-1 [43].

In keeping with current security best practices, the symmetric keys are padded according to the PKCS\#1-OAEP standard before encryption [44].

3 Results

3.1 Representative Cyber Physical System Analysis Results:

3.1.1 CAN Logging Design and Analysis

The performance evaluation was implemented by creating a CAN network consisting of the deterministic message generator and the CAN loggers under evaluation. The message generator generates 500 messages and forwards them onto the CAN network. The loggers record the messages with a time stamp. The Vector CANcaseXL logged all messages in order and preserved their content. Figure 29 illustrates that the vast majority of recorded messages mirrors the specified inter-message timing of 20 and 40 milliseconds, but a number of messages were recorded at +/-1 ms.

ACL #1: The first logging solution using only the Arduino UNO and CAN bus shield was not able to meet any of the previously defined logging requirements. While logging the first set of 500 messages the system began to drop messages. The time stamps indicate that the system could not cope with the rate of incoming data. Figure 30 illustrates that inter-message timings spiked several times above 200 ms. The data stream was completely compromised after approximately 150 messages.

ACL #2: This design logged all messages in order and preserved their content. Furthermore, the maximum inter-message timing offset was +/01 ms as shown in Figure 31. The ACL showed greater variation more frequently than the Vector logger. Nevertheless, logging all messages, preserving the order, and affecting the same inter-message logging delay makes this Arduino logger design a viable alternative to commercially available products in this scenario.

ACL #3: This design captures all messages, preserves the order and timing. Most of the messages perfectly mirror the inter-message timing of 20 and 40 milliseconds,

but, similar to the Vector, a few messages were recorded at +/-1 ms as shown in Figure 32.

3.1.2 Real time Replay Methodology: Logger Evaluation

The CAN replay evaluation of the loggers uses the real-time replay system to send out recorded CAN traffic and to compare the log files to reference data stored in the database for comparison. The order and completeness criteria established before must be met perfectly by any viable logging solution. The timing criteria do not have to be met perfectly, but must be appropriate for the purpose of the cyber-physical system analysis. The evaluation framework consists of the necessary CAN wiring harness/cables, the CompactRio platform with a NI 9853 module and the respective logger under evaluation.

Vector CANCase XL: This logger records all messages in order while preserving the message content. The inter-message timing is preserved very well with an average absolute inter-message timing error of approximately 7micro s. The average minimum absolute error timing is 0 micro s and the average maximum absolute error timing is approximately 298 microseconds. Considering the fact that the average inter-message timing of the reference data is approximately 1000 micro s, this error is acceptable for the continued system evaluation. Furthermore, the inter-message timing error does not indicate a slow/erroneous logging process since the underlying CAN network is, of course, considerably different from the one in the car that served as the data source. The overall inter-message timing error is depicted in Figure 33.

ACL #1: Not tested due to insufficiency of preliminary performance characteristics.

ACL #2: Testing revealed that this logger cannot cope with the volume of data and the transmission speed encountered during replay. The CAN shield receives all messages in the correct order but the system is not capable of concurrent, loss-less receiving, processing and logging. While the Toyota does not exhaust the transmission capacity of its CAN infrastructure, the Arduino Logger is unable to capture all messages during the evaluation and fails to meet the order and completeness characteristics with regard to the produced log file; the CAN shield meets the order and completeness characteristics but the log file output does not. Exhaustive optimization to the Arduino implementation, including using a binary log format, increased the logging accuracy to approximately 90%.

ACL #3: This system comprised of Arduino Due, Due CAN shield, and OpenLog chip initially encountered the same problems as the Arduino Uno solution. However, the Arduino Due features a USB port, which allows for much higher data transmission rates than standard serial ports. Using the USB port – as opposed to the serial port -- the Arduino Due equipped with the custom CAN shield met the order and completeness requirements during all 35 evaluations runs without any optimizations to the Arduino solution. The inter-message timing of the original CAN traffic is preserved well with an average absolute inter-message timing error of approximately 478 micro s. The average minimum absolute error timing is 0 micro s

and the average maximum absolute error timing is approximately 4512 micro s. The overall inter-message timing error is depicted in Figure 34.

3.1.3 Real-time Replay Methodology: System Characterization

Experimentation with the replay methodology applied CPS system characterization focusing on the voltage sensor of a Transmission Control Unit (TCU). Continuously recording the supply and the CAN voltage while manipulating the supply voltage at the source enables the assessment of the lag between both measurements. The test runs used in this analysis resulted in roughly 10000 CAN voltage readings recorded at 5 ms intervals and 100000 voltage supply measurements recorded at 500 micro s intervals. During the test run the voltage was initially held constant, then increased quickly, and held at the new level. Since both log files include a common tick count reading, the timing of the voltage change and the duration of the change can be extracted.

Figure 35 illustrates the changes in both voltages and also hint at the TCU voltage underreporting. The CAN voltage change lasts approximately 95 ms and covers a voltage differential of 1.056 V. The supply voltage change takes approximately 78 ms and the recorded voltage differential is 1.044 V. The TCU generated CAN voltage underreports the actual supply voltage very consistently by approximately 0.066 V.

The final aspect of system characterization explored was the potential for simulation and corresponding analysis. Figure 36 shows that the simulated system matches the TCU output behavior extremely well, resulting in very little differential. The voltage differential between the two systems averaged over approximately 47,000 measurements is 0.000614 V with a maximum differential of 0.065 V and a minimum of 0 V. Since the conversion delay of 5 ms and the underreporting of 0.066 V were integrated during the design of the simulated TCU, these parameters are deemed confirmed.

3.1.4 Formal Verification Study

The Keymaera system using hybrid programs was selected as the candidate formalism and tool set to apply to vehicle CPS. Investigative modeling and analysis was pursued with the hypothetical voltage control system presented in Figure 37. The control system must keep the supply voltage between 10 V and 12 V. Once started, the system enters state “ON" and the voltage, initially set to 10 V, grows at increments of 0:1 V. The system stays in “ON" state as long as the voltage is <=12 V. However, once 11:5 V is reached, the system may move into state “OFF". Enabling the transition at 11:5V instead of 12V gives the system a chance to transition without violating the conditions of state “ON". In state “OFF" the supply voltage wanes at 0:2V decrements, potentially down to 10.4V. Once the voltage drops below 11V, the system has the option to transition to state “ON". During the transition from “ON" to “OFF" the voltage increases by 0:1 V, during the transition from “OFF" to “ON" the voltage decreases by 0:5 V . This model can be translated into a Keymaera source file written in the Key language. Once loaded, Keymaera analyzes the specified system and attempts to verify that the supply voltage it guaranteed to

remain between 10 V and 12 V. However, the transition from “OFF" to “ON" has the potential to drop the voltage below 10 V and Keymaera is therefore unable to prove the underlying assumption.

Figure 37 shows a modified voltage control system whose source code is available in Appendix D. The modified control system features slightly changed state transitions that allow Keymaera to prove that the supply voltage remains within the specified boundaries. The actual output of the validation is shown in Figure 38.

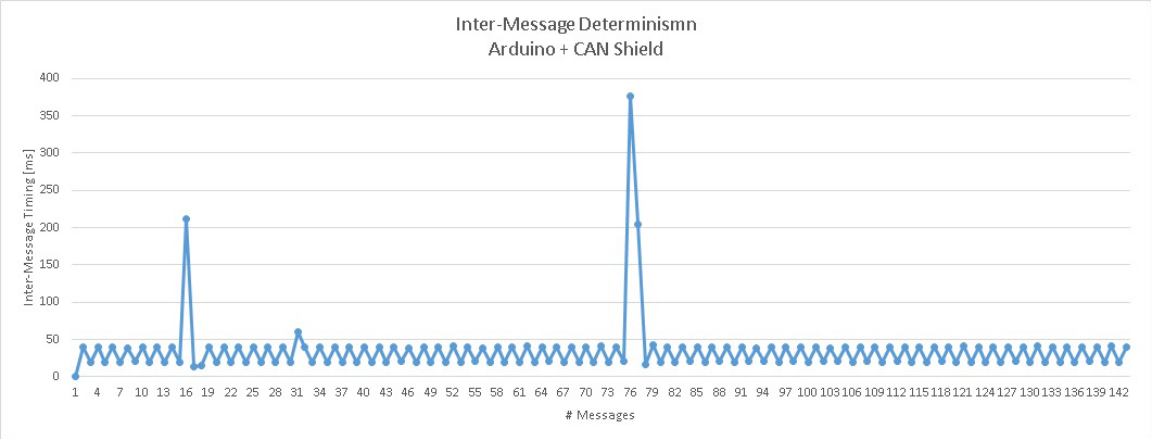


Figure 29: Arduino with CAN Shield Logging Results.

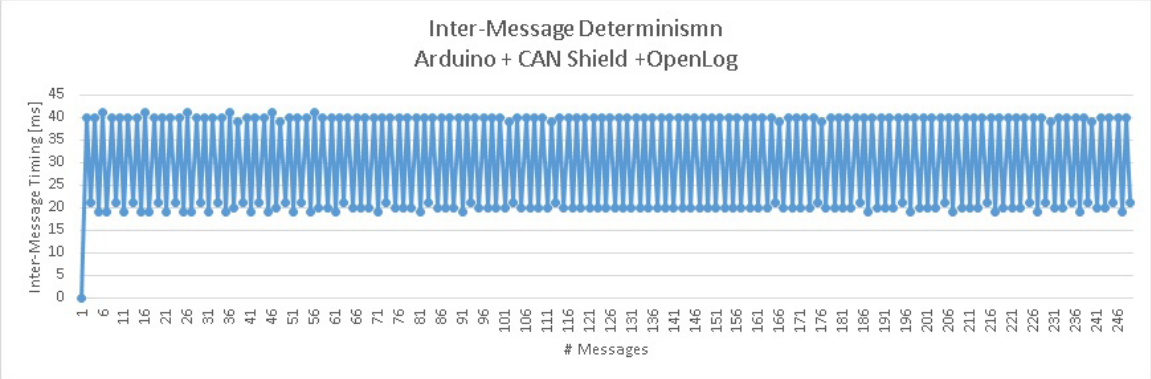


Figure 30: ACL #2 Logging Results.

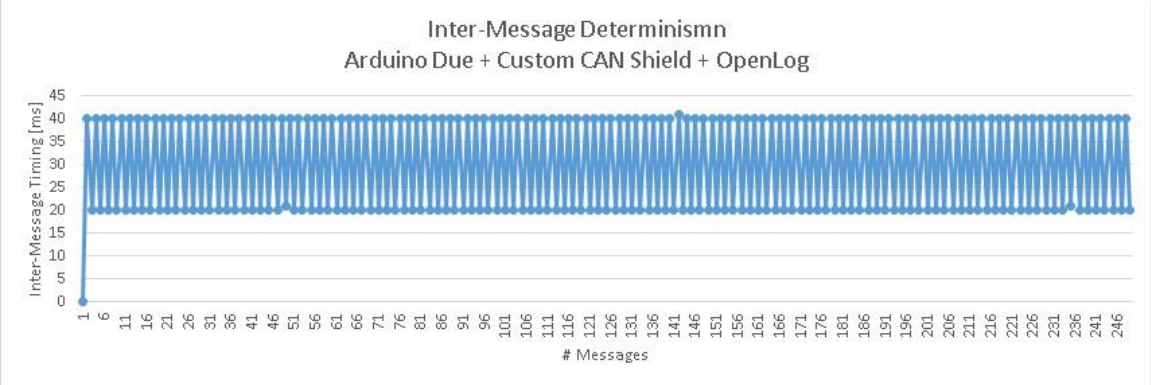


Figure 31: ACL #3 Logging Results.

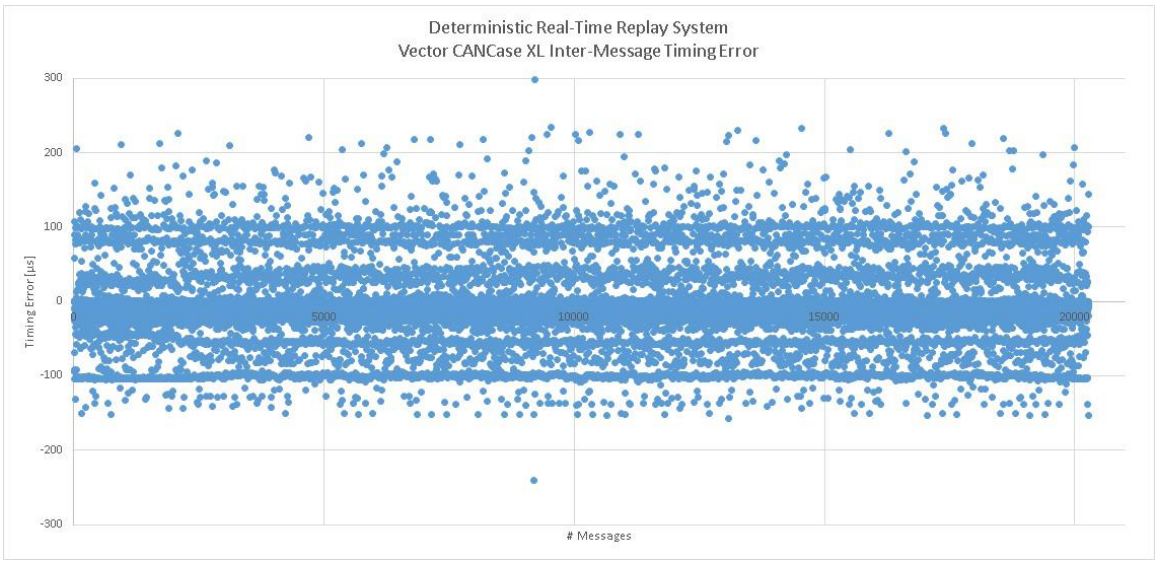


Figure 32: Vector CANCase XL Inter-message Timing Error.

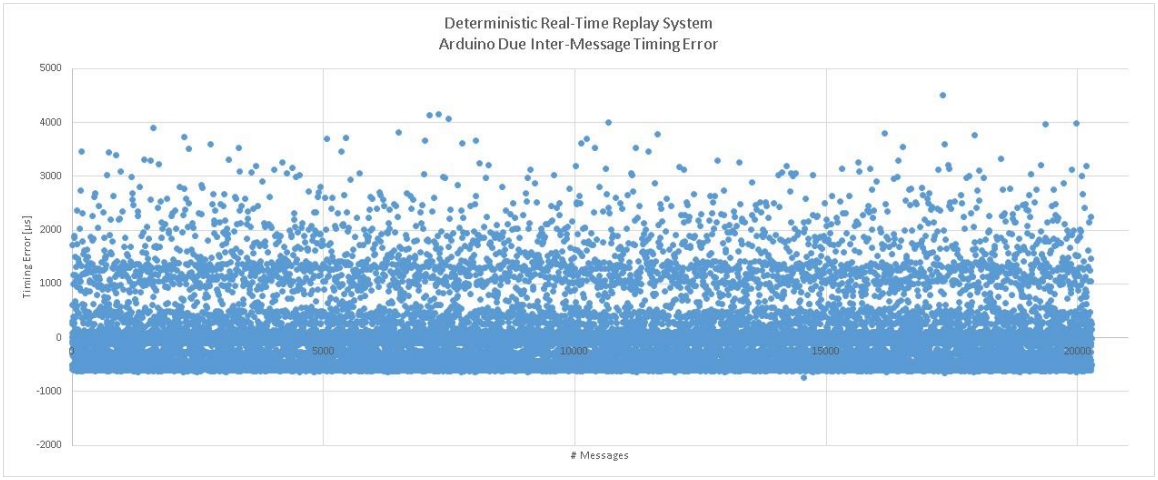


Figure 33: ACL #3: Inter-message Timing Error.

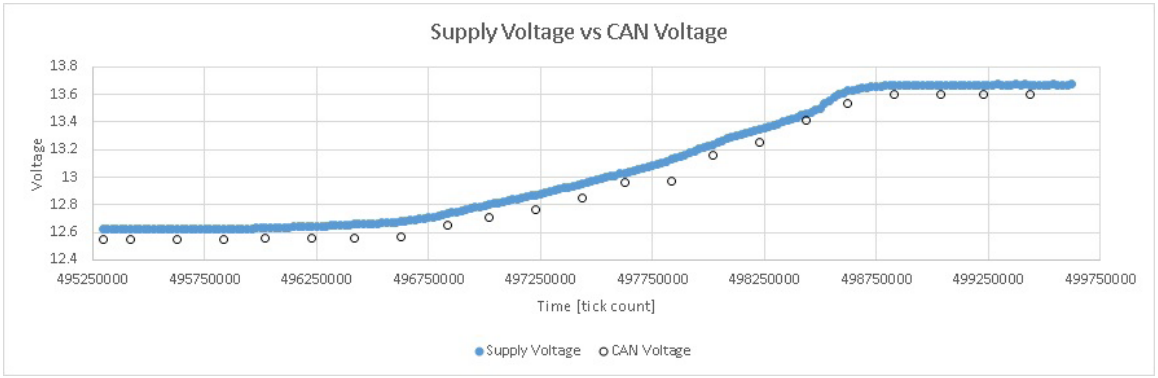


Figure 34: Supply/CAN Voltage Comparison.

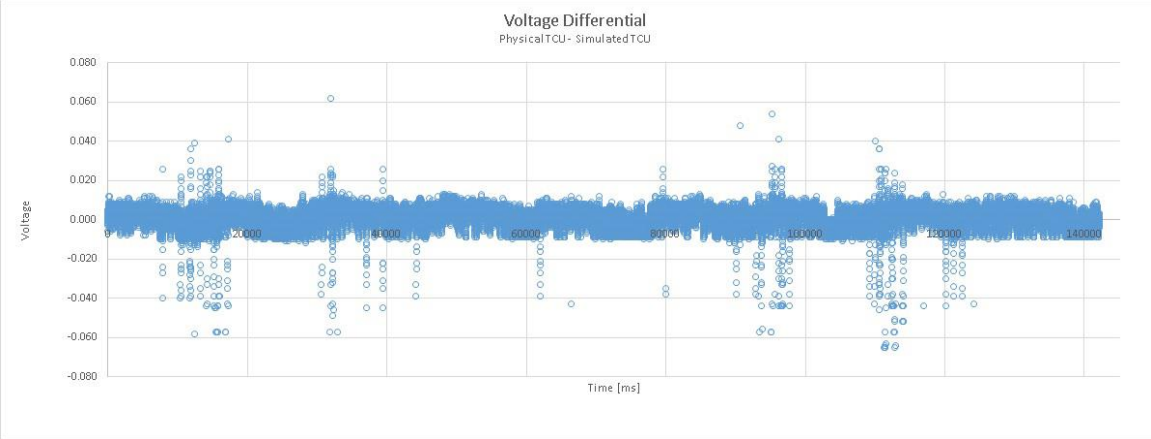


Figure 35: TCU vs. Simulated TCU Voltage Differential.

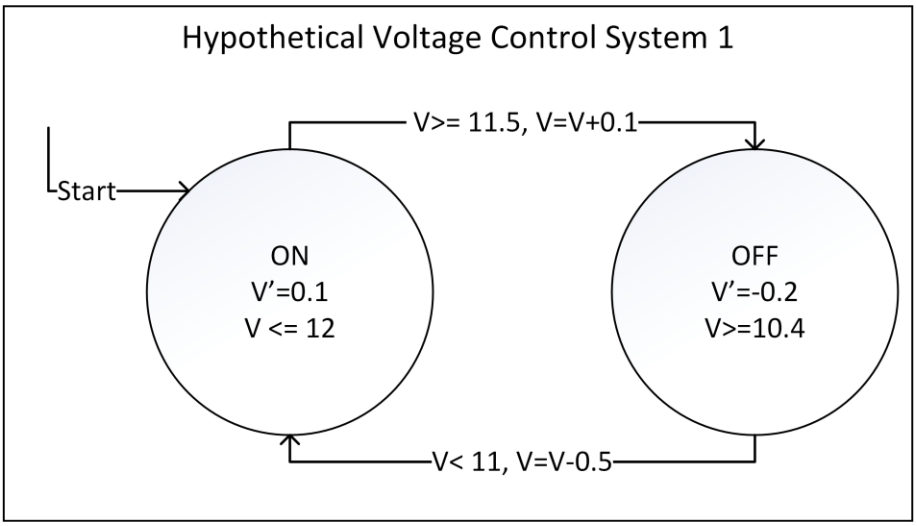


Figure 36: Keymaera Voltage Control System 1.

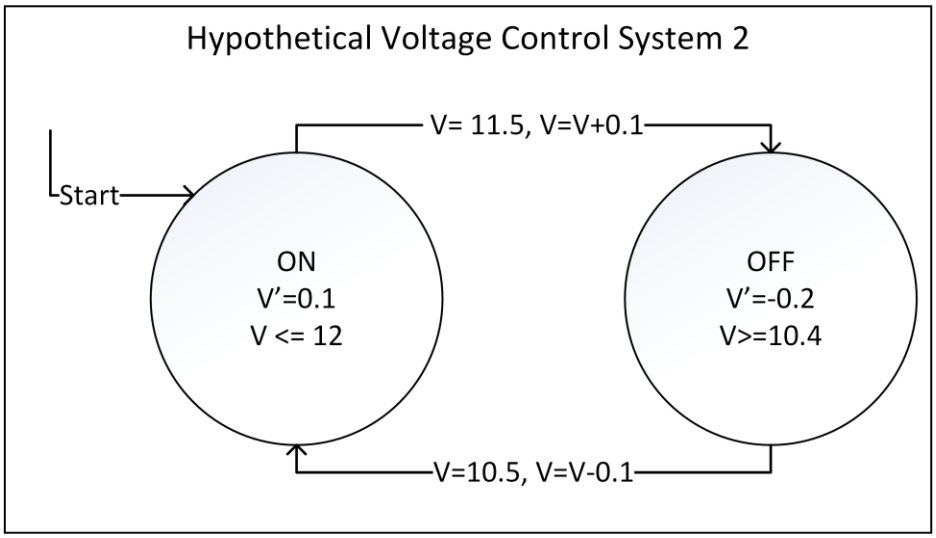


Figure 37: Keymaera Voltage Control System 2.

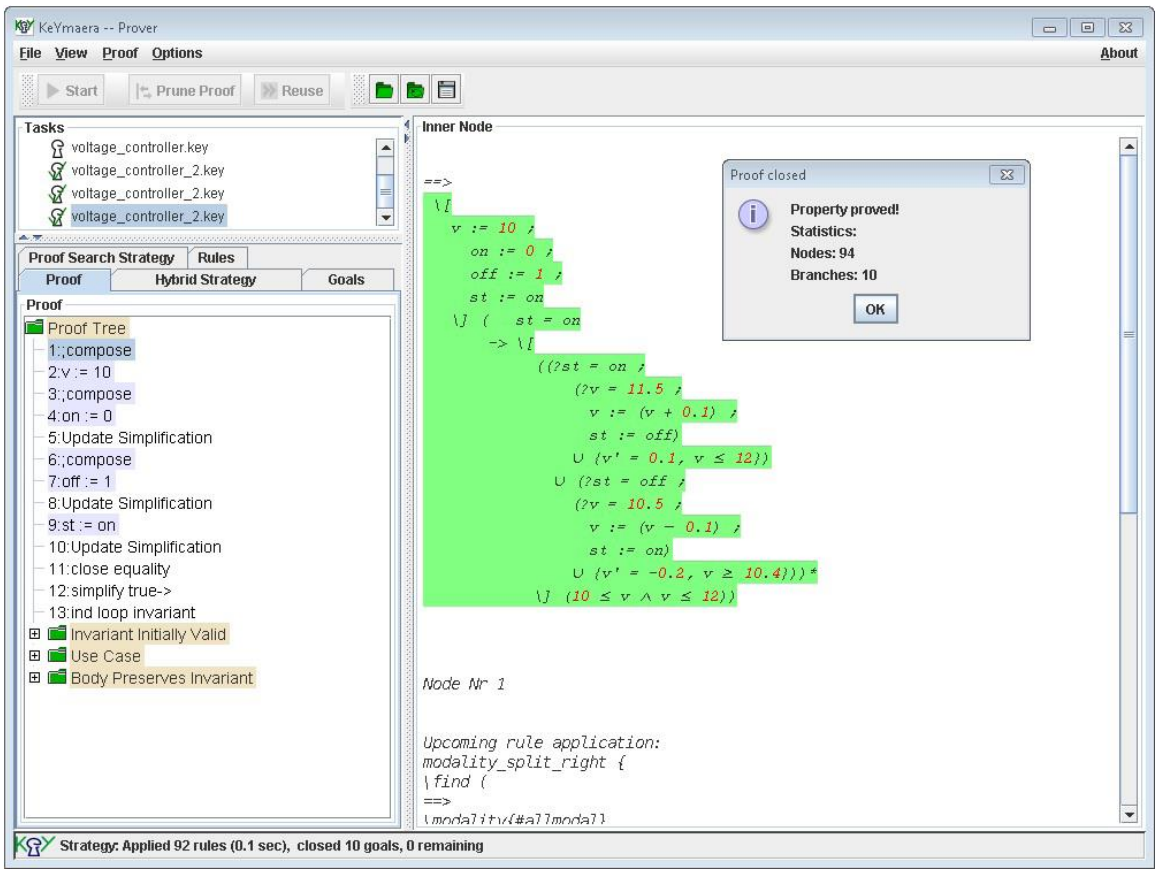


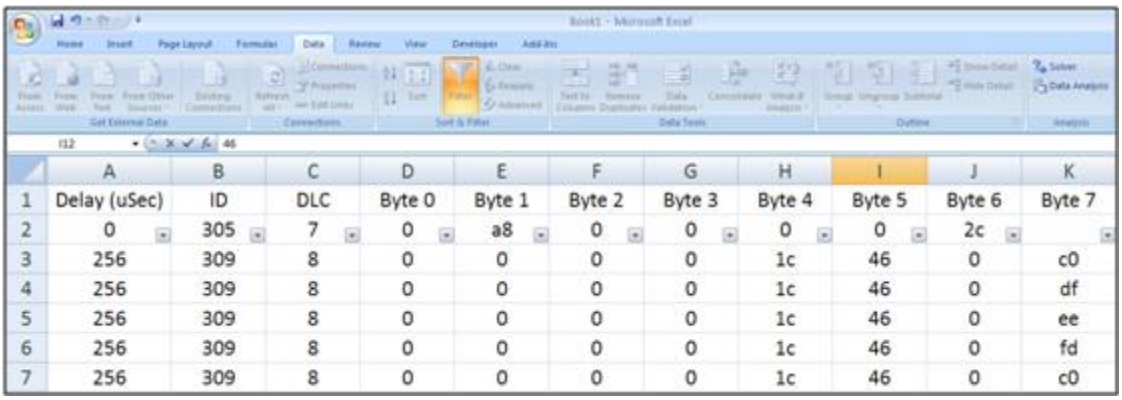
Figure 38: Keymaera Voltage Control System Validation Result.

3.2 Passenger EDR Analysis Application to 2012 Honda Vehicles

The deterministic CAN replay system was used to evaluate 2012 Honda CR-V and Civic SRS modules. This section will explain the study of the accuracy of the EDR speed and steering data as well as the EDR transfer functions.

3.2.1 4.1 Identification of SRS Sources

It is important to know which message IDs are sourcing the information to the SRS module. To determine these messages, the data within a replayed CAN stream were set to a specific value and examined on the CDR report. This was done by changing the values of the byte(s) responsible for the SRS data. The CAN files were altered by splitting the messages into bytes, filtering by ID, and changing the desired byte(s). Figure 39 shows the CAN file split into bytes in columns D-K, and filtered by ID, which is shown in Column B.



	A	B	C	D	E	F	G	H	I	J	K
1	Delay (uSec)	ID	DLC	Byte 0	Byte 1	Byte 2	Byte 3	Byte 4	Byte 5	Byte 6	Byte 7
2	0	305	7	0	a8	0	0	0	0	2c	
3	256	309	8	0	0	0	0	1c	46	0	c0
4	256	309	8	0	0	0	0	1c	46	0	df
5	256	309	8	0	0	0	0	1c	46	0	ee
6	256	309	8	0	0	0	0	1c	46	0	fd
7	256	309	8	0	0	0	0	1c	46	0	c0

Figure 39: Screenshot of CAN message identification verification procedure.

ID 0x309 bytes 4 and 5, column H and I respectively, were set to constant value of 0x1c46 as shown in Figure 39. The modified CAN file was replayed to the recording SRS module and the corresponding Bosch CDR report was generated as shown in Table 6. The vehicle indicated speed remained constant in Table 6 where the baseline data shown in Table 5 varied from 11 to 42 km/h. The altered file's speed remaining constant as opposed to the variable speed report for the un-altered CAN report definitively identifies ID 309 bytes 4 and 5 as the SRS Vehicle Indicated Speed source. Additionally, 0x1c46 has a decimal value of 7238, which represents the number of 0.01 km/h increments, or a speed of 72.38 km/h. This verifies bytes 4 and 5 of CAN ID 0x309 are responsible for the SRS indicated vehicle speed pre-crash data.

Table 5: Pre-crash data from the baseline CAN data replayed to the SRS module.

Pre-Crash Data -5 to 0 sec [2 samples/sec] (Event Record 1)
(the most recent sampled values are recorded prior to the event)

Time Stamp (sec)	Speed, Vehicle Indicated (MPH [km/h])	PCM Derived Accelerator Pedal Position, % full	Service Brake (On, Off)	ABS Activity (On, Off)	Stability Control (On, Off, Engaged)	Steering Input (deg)	Engine RPM	Accelerator Pedal Position, % full
-5.0	7 [11]	19	Off	Off	On Non-Engaged	-35	1,600	19
-4.5	8 [13]	25	Off	Off	On Non-Engaged	85	1,800	25
-4.0	9 [15]	28	Off	Off	On Non-Engaged	130	1,900	28
-3.5	11 [17]	48	Off	Off	On Non-Engaged	130	2,200	48
-3.0	12 [20]	98	Off	Off	On Non-Engaged	115	2,700	98
-2.5	15 [24]	100	Off	Off	On Non-Engaged	85	3,000	100
-2.0	18 [29]	100	Off	Off	On Non-Engaged	30	3,300	100
-1.5	21 [33]	100	Off	Off	On Non-Engaged	20	3,700	100
-1.0	23 [37]	100	Off	Off	On Non-Engaged	10	4,100	100
-0.5	26 [42]	100	Off	Off	On Non-Engaged	5	4,500	100
0.0	26 [42]	100	Off	Off	On Non-Engaged	5	4,500	100

Table 6: ID 0x309 Bytes 4 and 5 were set to 0x1c46, which corresponds to 72.38 km/h.

Pre-Crash Data -5 to 0 sec [2 samples/sec] (Event Record 1)
(the most recent sampled values are recorded prior to the event)

Time Stamp (sec)	Speed, Vehicle Indicated (MPH [km/h])	PCM Derived Accelerator Pedal Position, % full	Service Brake (On, Off)	ABS Activity (On, Off)	Stability Control (On, Off, Engaged)	Steering Input (deg)	Engine RPM	Accelerator Pedal Position, % full
-5.0	45 [72]	19	Off	Off	On Non-Engaged	-95	1,700	19
-4.5	45 [72]	24	Off	Off	On Non-Engaged	65	1,700	24
-4.0	45 [72]	28	Off	Off	On Non-Engaged	120	1,900	28
-3.5	45 [72]	36	Off	Off	On Non-Engaged	125	2,100	36
-3.0	45 [72]	80	Off	Off	On Non-Engaged	115	2,600	80
-2.5	45 [72]	100	Off	Off	On Non-Engaged	95	2,900	100
-2.0	45 [72]	100	Off	Off	On Non-Engaged	45	3,200	100
-1.5	45 [72]	100	Off	Off	On Non-Engaged	25	3,600	100
-1.0	45 [72]	100	Off	Off	On Non-Engaged	15	4,000	100
-0.5	45 [72]	100	Off	Off	On Non-Engaged	5	4,400	100
0.0	45 [72]	100	Off	Off	On Non-Engaged	5	4,600	100

Furthermore, bytes 4 and 5 of ID 0x309 were set to a value of 0x1c64 or 7268 (72.68 km/h) to determine if the SRS module truncates or rounds the speeds. After this change the vehicle indicated speed remained 72 km/h indicating the module truncates the decimals of the CAN speed record. Bytes 4 and 5 were then set to 0x1c83 or 7310 (73.1 km/h) which generated a record of 73 km/h, indicating that the reported speed does not round to the nearest even km/h, but rather reports only the CAN speed integer value. With the data processing algorithm established, an accuracy assessment can commence by truncating CAN speed and comparing it to the external reference speed.

To determine the steering translation, the Civic was taken through a series of lock-to-lock turning maneuvers while CAN data was recorded. These maneuvers produced maximum and minimum steering inputs of nearly ±570° at the steering wheel. The CAN data was decoded using a signed 16-bit integer. The lock points

along with (CAN decimal =0, steering angle =0°) were plotted and fit with a line to determine the value of the least significant bit (LSB) of the CAN message for steering. The resulting raw data is shown in Figure 40. All CAN IDs used for different pre-crash data are shown in Table 7.

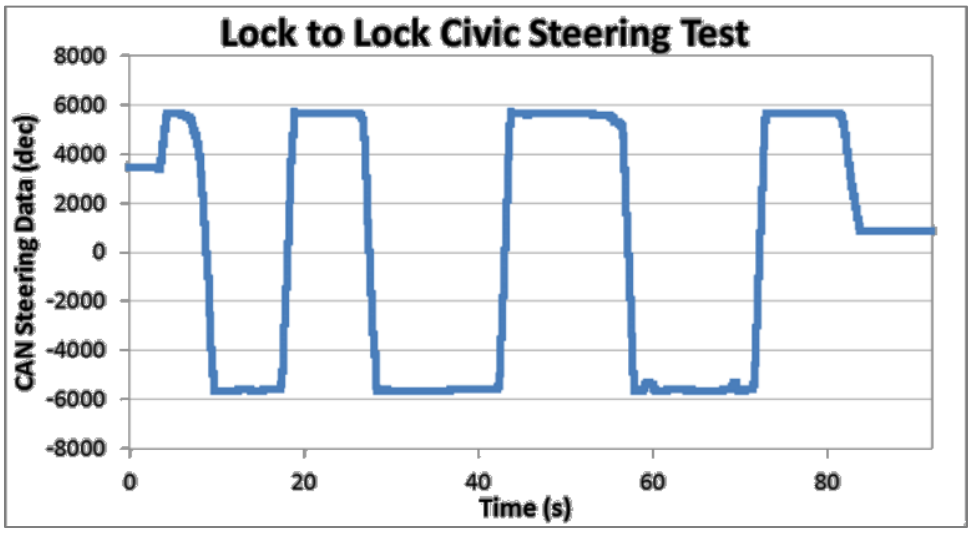


Figure 40: Lock-to-lock Civic steering test data.

Table 7: SRS CAN ID Data Source

Quantity	CAN ID	Byte(s)	Likely Conversion Method	CAN Refresh Rate (s)
Speed Vehicle Indicated	0x309	4 and 5	0.01 km/h per LSB	0.1
Accelerator Pedal Position	0x17c	0	0.5% per LSB	0.01
Engine RPM	0x17c	2 and 3	1 rpm per LSB	0.01
Service Brake	0x17c	Bit 0 of Byte 4	1 = On, 0 = Off	0.01
Steering Wheel Angle	0x156	0 and 1	-0.1 degree per LSB, signed integer	0.01

The Civic front wheels were also placed on angle measuring plates and the steering wheel was turned in 90 degree increments (as measured by a level) while monitoring the CAN bus steering angle parameter. The CAN bus data accurately reflected the steering wheel inputs. Honda EDR data limitations report that the EDR reports steering angle with a resolution of 5 degrees and rounds CAN bus data to the nearest 5 degrees. To test this, the SRS steering source, CAN ID 0x156 bytes 0 and 1, was set to three constant values: 12.9, 14.7, and -14.8 degrees. When these values were broadcast, the SRS reported steering inputs of 10, 10, and -10 degrees

respectively. These results suggest that the SRS does not round the steering value, but truncates it. The results of these tests are summarized in Table 8

Table 8: SRS Steering Truncation Test Results

Steering Broadcast (deg)	Byte 0 and 1 Corresponding Hex	SRS Reported Steering (°)	Rounded Steering (°)
12.9	FF7F	10	15
14.7	FF6D	10	15
-14.8	0094	-10	-15

3.2.2 Passenger Car EDR Speed Accuracy

The speed message, 0x309 bytes 4 and 5 (counting from zero), was also perceived to track with the display on the digital speed indicator in the instrument cluster. The graph shown in Figure 41 demonstrates that the indicated vehicle speed can be nearly 0.8 seconds late in reporting the value. The front left wheel speed found from the message 0x1D0 bytes 0 and 1 show that wheel speed tracks the VBOX speed appropriately during a tire slip with the ABS system engaged. Based on this observation and the fact that the data were synchronized using the CAN bus, the timing delays found in message 0x309 are real. Furthermore, the indicated vehicle speed message updates every 0.1 seconds but changes value every 0.6 seconds in this test. Therefore, it is expected that the Honda Civic will likely have a repeated data point on the 0.5 second intervals shown in the EDR records. However, not all data gathered show the 0.6 second wait to change, which suggests there may be some other processing in the computer that transmits the message that takes priority over updating the indicated speed.

The data shown in Figure 42 shows the CR-V indicated speed updates and changes every 0.1 seconds. The speedometer on the CR-V used a needle as opposed to a digital display. The wheel speed signal drops as the braking commences and periods of higher slip are shown. The indicated speed tends to follow a subdued path when the wheel speed drops, indicating an averaging effect for the indicated speed. With no wheel slip, the indicated vehicle speed closely matches the VBOX speed.

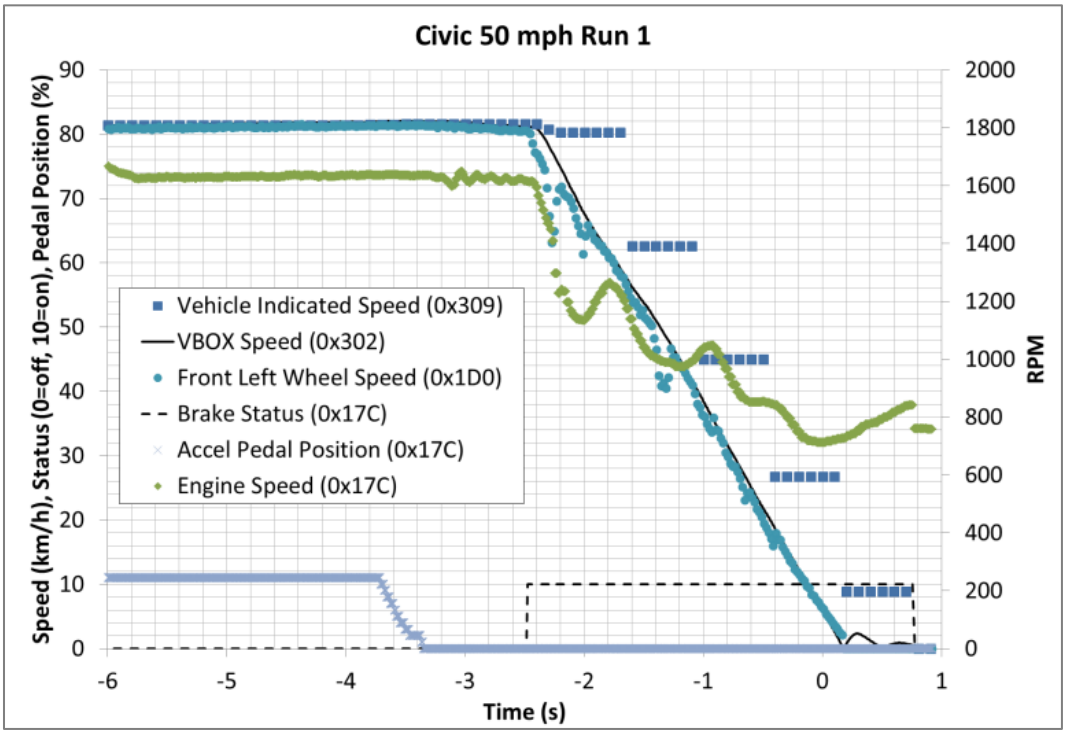


Figure 41: Civic hard brake data from 50 mph.

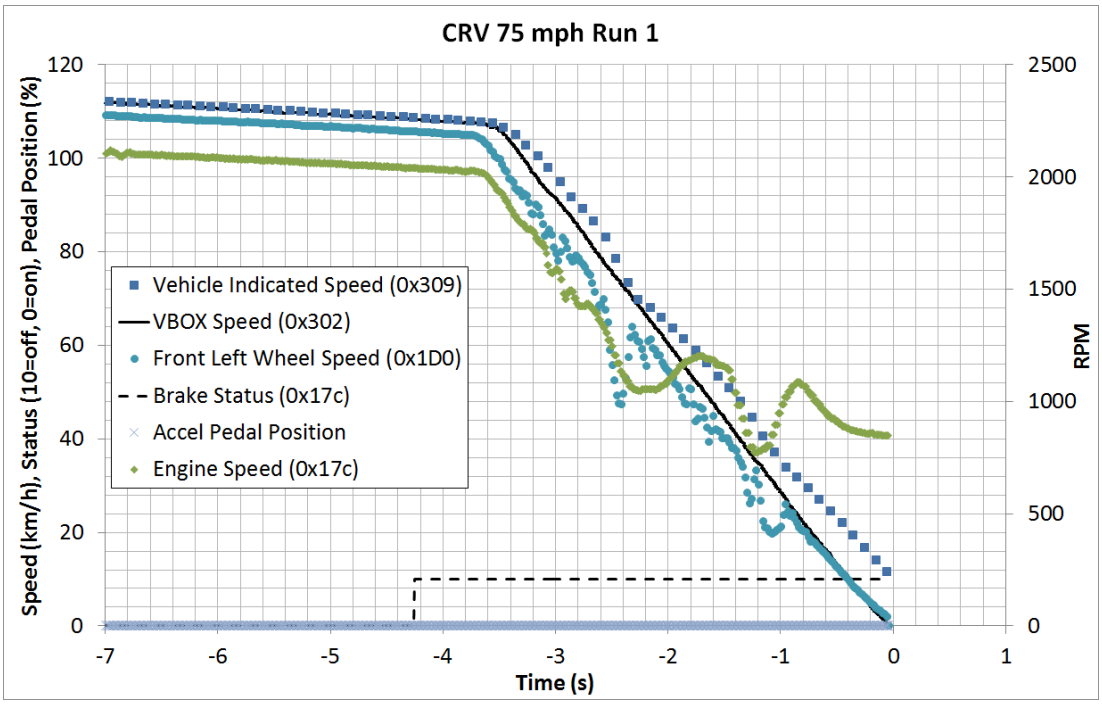


Figure 42: CR-V Hard-brake from 75mph

3.2.2.1 2012 CR-V Steady State Test

An instrumented 2012 Honda CR-V was driven on an expressway and 3 minutes of CAN traffic for normal driving was recorded. Since the VBOX data was transmitted on the CAN, the time synchronization of data was automatic. The speed record from the VBOX and the indicated vehicle speed (0x309) are shown as lines in Figure 43. Since the VBOX transmits data at 100 Hz and the indicated speed is updated at 10 Hz, the VBOX speed signals were smoothed using a moving average and resampled to align with the less frequent CAN speed. This reduction by a factor of 10 resulted in 1800 messages for comparison. Having learned the CAN bus to EDR transfer function, the indicated vehicle speed (0x309) was truncated to the next lower whole km/h to reflect the data that would be recorded in the SRS. This would produce an expected error band of 0 to 1 km/h when the error is defined as the GPS speed – EDR speed. This also allows a large sample size in comparison to what could be achieved with actual EDR recordings.

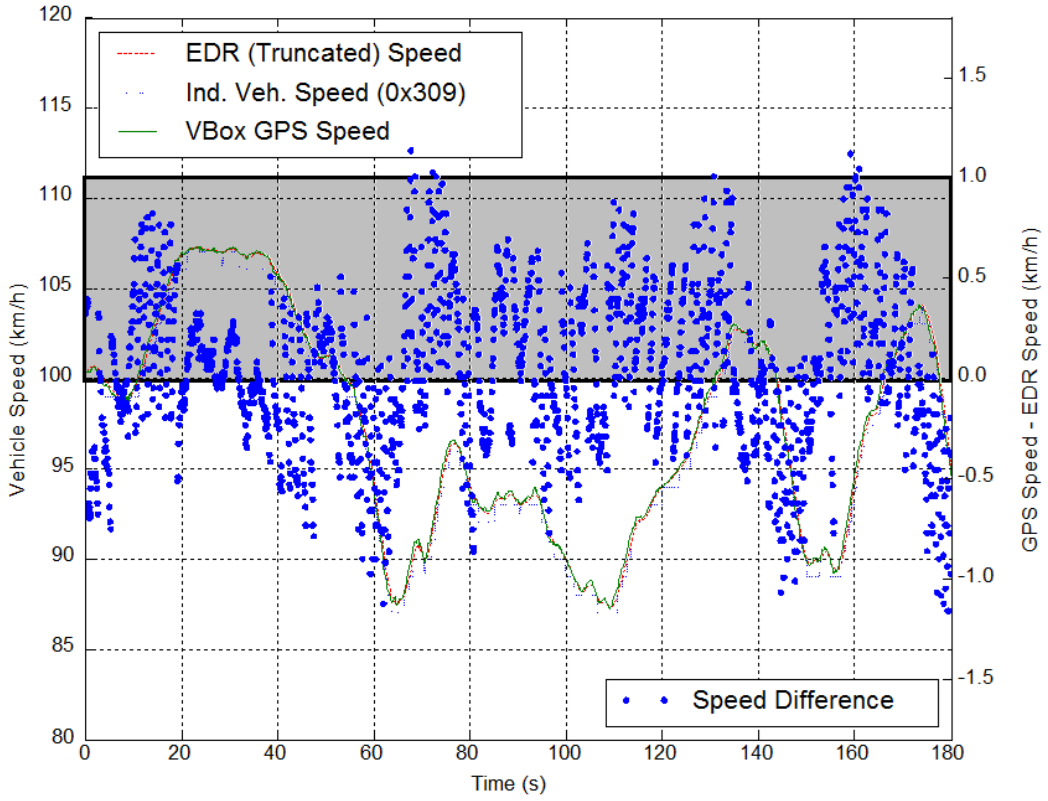


Figure 43: Speeds and speed differences for normal highway driving with a 2012 Honda CR-V. The gray band indicates expected error bounds from data truncation.

To assess the accuracy of the speed data, the differences between the GPS speed and the EDR theoretical speed were determined and plotted against the right axis of Figure 43. The gray box behind the figure show the theoretical error bound from truncation alone. The differences were between +/- 1.15 km/h with a mean of 0.033

km/h and standard deviation of 0.39 km/h. This suggests that the CR-V normal driving speed data is accurate to about 1%.

3.2.2.2 2012 Civic Steady State Test

To assess the steady state accuracy of the EDR in the Honda Civic, the Civic was driven starting at 80 km/h (50mph) on speed control and the speed control was incremented by 1 mph approximately every 4 seconds up to a speed of 113 km/h (70 mph). The vehicle had time to stabilize in between increments and the acceleration to the next higher speed was gradual enough so as not to produce any significant wheel slip. The CAN bus vehicle indicated speed was truncated to the next lower whole km/h, which is the value that would be recorded in the EDR, and the difference between the VBox GPS signal and the EDR value was calculated. These difference data are plotted against the right axis of Figure 44.

For the 975 data points recorded, the mean difference was +0.22 km/h with a standard deviation of 0.53km/h. The maximum difference ranged from -1.38 to +1.95 km/h. Figure 44 plots the data for the VBOX GPS, the CAN bus vehicle indicated speed, and the truncated speed that would be recorded by the EDR. Differences between GPS and EDR are plotted as points relative to the scale on the right side axis. The gray box shows the theoretical error limits from truncation only. There was no evidence that the error was dependent on speed over the 80 to 113 km/h range.

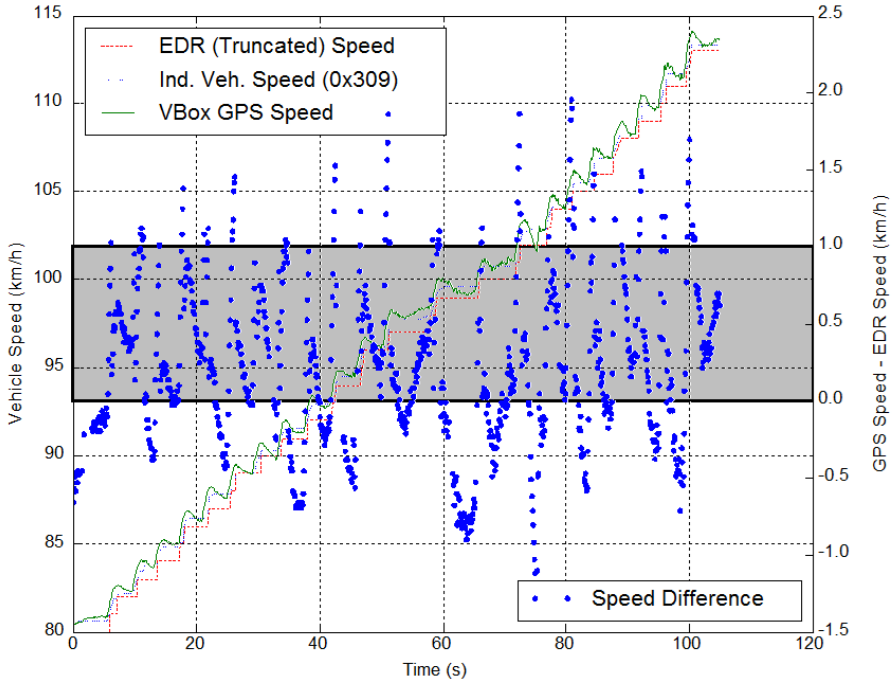


Figure 44: Speeds and speed differences for driving by gradually incrementing the cruise control with a 2012 Honda Civic. The gray band indicates expected error bounds from data truncation.

3.2.2.3 2012 CR-V Accuracy During Maximum ABS Braking

The CAN bus data was played back to the ACM and events were set with the non-deployment apparatus. The start time of the CAN bus file was incremented by 0.1 seconds each run for 10 runs to see the differences in the EDR data. When that was completed the same CAN file was played with the same timing five times in a row to demonstrate repeatability.

Both the recorded CAN data and the Bosch CDR reported data from the 2012 Honda CR-V are shown in Figure 45 through Figure 48. These graphs represent the CAN data from driving tests and the acceleration data from the tests on the non-deployment setting device.

The pre-crash information for two different test runs is shown in Figure 45 and Figure 47. The data in these graphs show the VBOX 3i as a solid line. The green plus symbols represent the wheel speed for the left front wheel. Only a single wheel speed is displayed; if multiple wheels are displayed the graph becomes too busy. When the brakes were first applied, the wheel speed trace shows a sharp reduction until the ABS system intervenes to relieve the brake pressure and allow the wheel to rotate with a controlled slip. Wheel slip causing under reporting of ground speed has been well documented in the literature (e.g. [20]) and will be acknowledged but not analyzed extensively in this study. The interpretation of the wheel speed message seems to slightly under-report the VBOX speed signal; however, the messages for wheel speed slightly lead the VBOX signal in time. The blue diamonds

represent the indicated vehicle speed, which is the source for the EDR data. In all cases, the diamonds must lead the squares representing the CDR reported speed values. Since the EDR functionality truncates the speed values, the data must be below the corresponding indicated speed message. The pre-crash graphs also show the number of hundreds of RPM the engine was turning. The solid line represents the CAN message value and the circles represent the RPM reported by the CDR tool, which are truncated to the nearest 100 RPM. Finally, the acceleration trace from the accelerometer mounted on the SRS sled. The acceleration trace enables synchronization of the crash data to the pre-crash data and the establishment of t0.

Based on the data shown in Figure 45 and Figure 47, the SRS module may over report vehicle indicated speeds during hard braking by as much as 10 km/h due to reporting delays. The time delay can be seen where the hard brake corner extends beyond that of the VBOX trace slightly. The CDR reported data for the CR-V is also delayed, but not consistently. The delay from the EDR function can be examined by repeating tests with the apparatus for this study.

The external accelerometer traces in Figure 45 and Figure 47 are examined with a smaller time scale in Figure 46 and Figure 48, respectively. Effectively zooming in on the non-deployment event enables analysis of the delta-V and acceleration data reported in the Bosch CDR report. Figure 4.8 and Figure 4.10 show the raw accelerometer data in g's sampled at 4000Hz as a solid line. The acceleration from the CDR report is represented by the squares. The Data Limitations section of the CDR report defines t0 as when a change in cumulative delta-V of -0.8 km/h over 0.020 seconds occurs. The location of t0 corresponds to the zero mark on the time axis. To determine the delta-V, the previous 20 ms of the accelerometer signals were integrated (summed and multiplied by the sampling period) in-place and converted to units of km/h. The result of the respective Delta-V calculations is represented by the broken lines in Figure 46 and Figure 48 and trend with the CDR reported delta-V.

Once t_0 was established, a cumulative delta-V was calculated starting at t_0. This calculation is shown as the blue dashed line in Figure 46 and Figure 48. The corresponding CDR reported delta-v is represented as green circles on the graphs. According to the Data Limitations, the recording of delta-V stops 30 ms after the event is over (when the delta-V changes by less than -0.8 km/h in 20ms. Examination of the record around 0.045 seconds shows the trace representing delta-V from the previous 20 ms rises above -0.8 km/h. Therefore, 0.030 seconds or 3 samples more of recorded delta-V data are shown before reporting zeros, as expected.

The data plotted in Figure 46 and Figure 48 is similar, yet unique. The external accelerometer has a similar trace in each run but there is enough variation to show different data samples on the CDR report. Since the external accelerometer and the SRS accelerometer are different and the filtering mechanism and internal sampling of the SRS accelerometer is not known, the accelerometer and delta-V data will likely never match perfectly; however, the trends and patterns between the two accelerometers correlate. This is important because it gives confidence to the

interpretation of the t0 mark and the assessments of any timing observations are well founded.

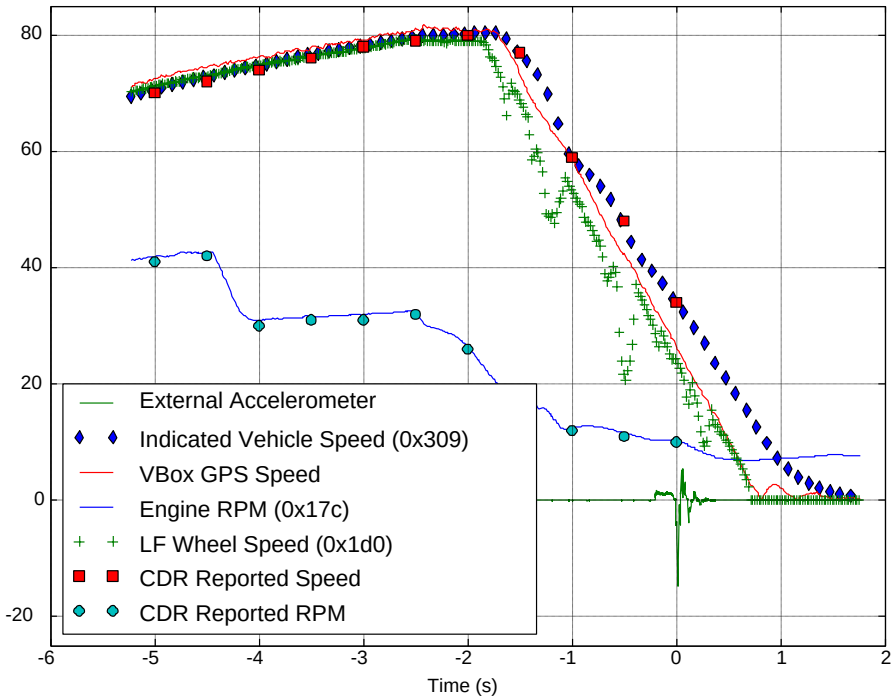


Figure 45: Graph of the Pre-Crash data from the CDR report with the CAN messages for a Honda CR-V at city street speed.

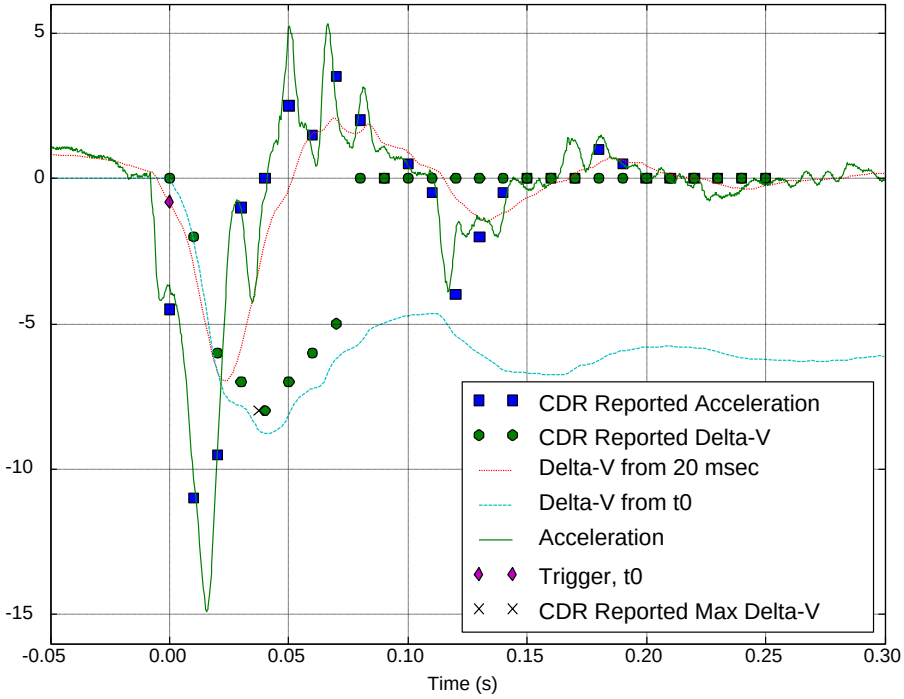


Figure 46: Graph of the crash data corresponding to the CR-V data shown in Figure 45.

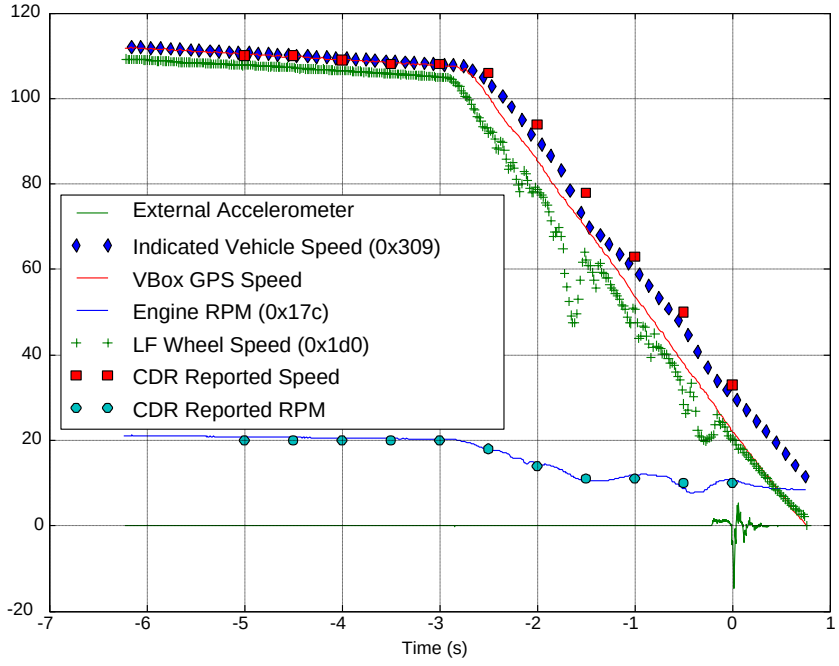


Figure 47: Graph showing pre-crash data for a Honda CR-V during hard braking at highway speed.

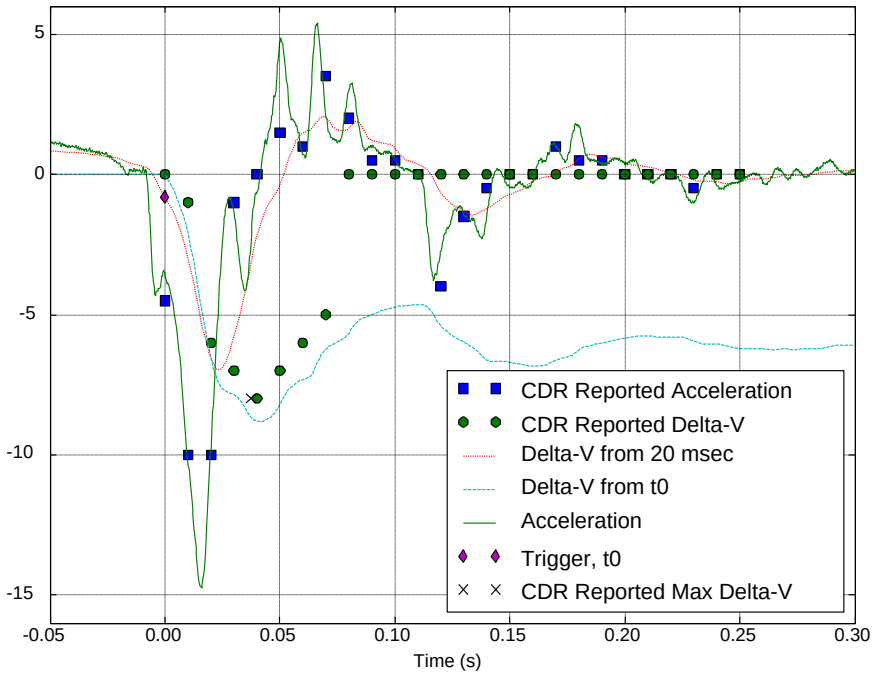


Figure 48: Graph of the crash data corresponding to the CR-V data shown in Figure 47.

3.2.2.4 2012 Civic Accuracy During Maximum ABS Braking

Figure 49 and Figure 50 show a typical test run braking from 80km/h (50mph) and Figure 51 and Figure 52 show the results of braking from 113 km/h (70 mph). The odd number figures show the speeds and RPM versus time, the even numbered figures show the details of the event triggering similar to the discussion on the CR-V above. Note that the CAN bus vehicle indicated speed values repeat six times in a row, finally changing at 0.6 second intervals. The resulting increased delays (versus the CR-V) in the EDR reporting lead to over-reporting speed by as much as 20 km/h. During maximum braking the reported speed should decrease with each new data point, but in some cases an old speed value is repeated giving the false impression that no vehicle speed reduction has occurred over the interval. The 0.6 second change interval was consistent during the maximum braking runs conducted, but in other tests involving normal driving the Civic updated more frequently.

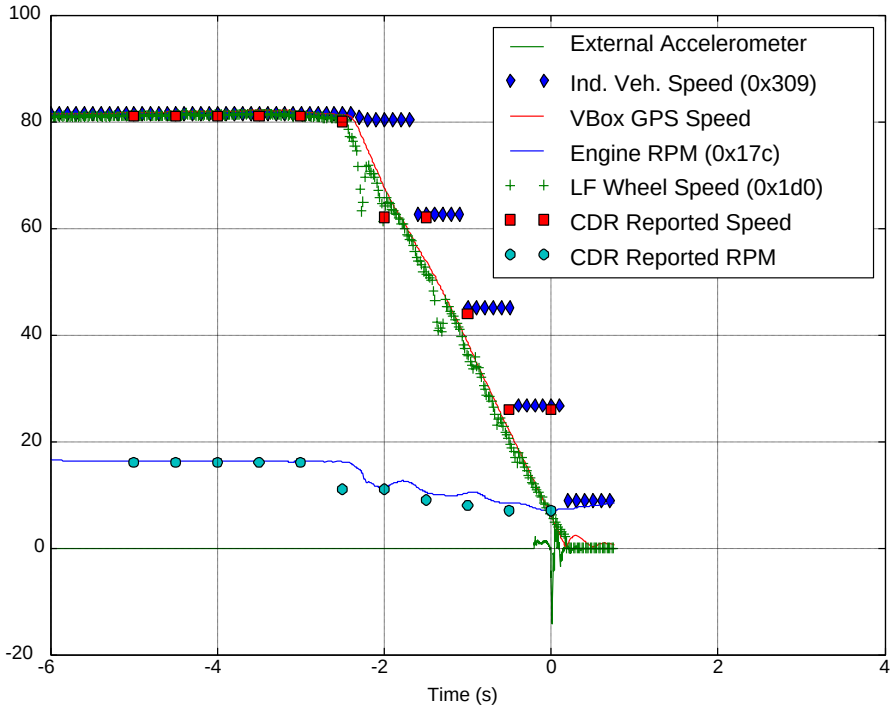


Figure 49: Civic pre-crash data braking from 80 km/h (50 mph).

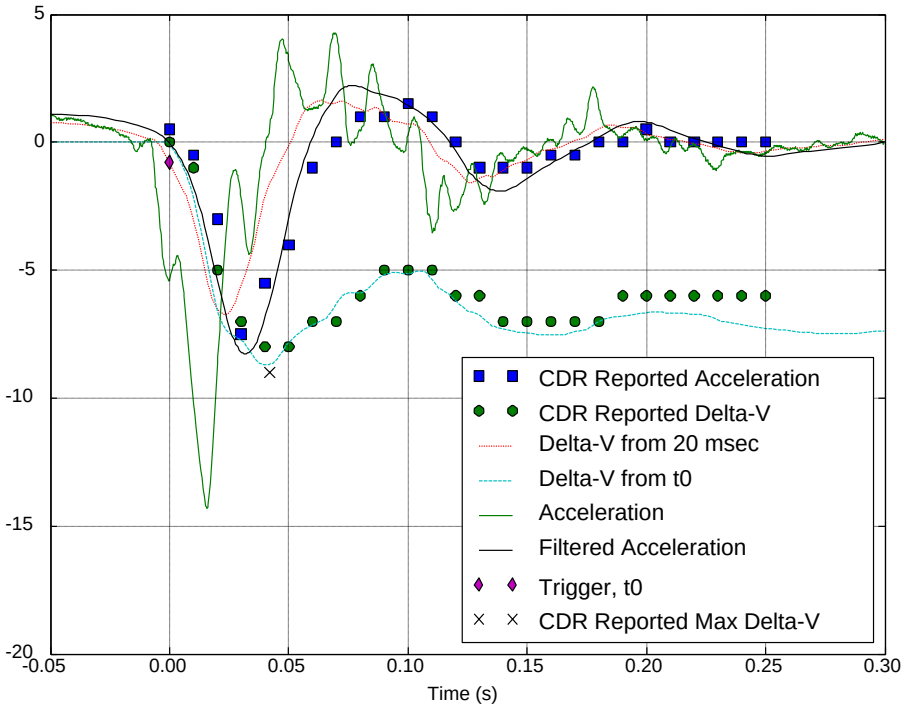


Figure 50: Crash data for the Civic corresponding to Figure 49.

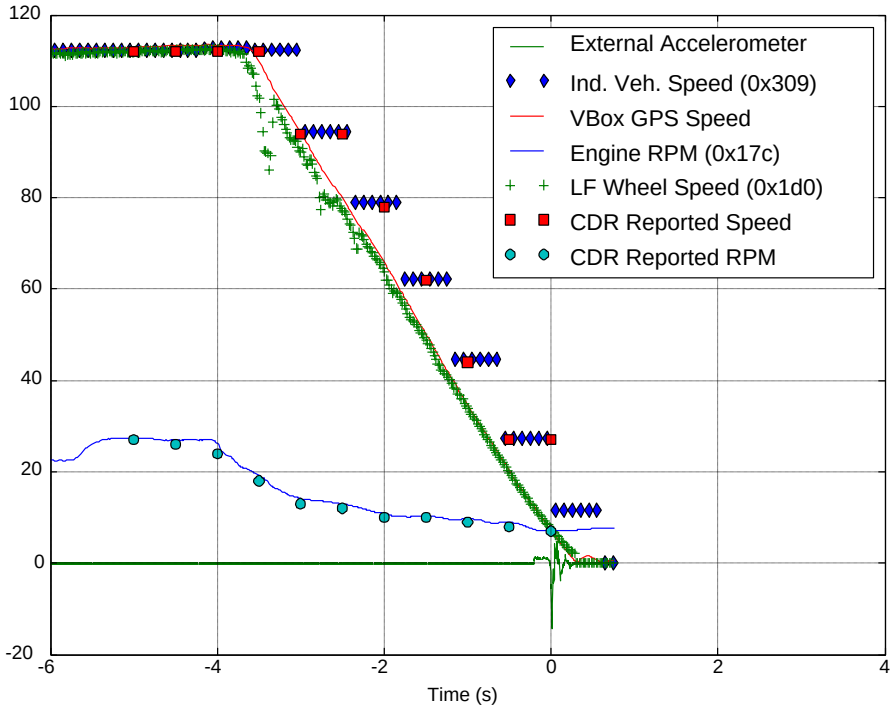


Figure 51: Civic pre-crash data braking from 113 km/h (70 mph).

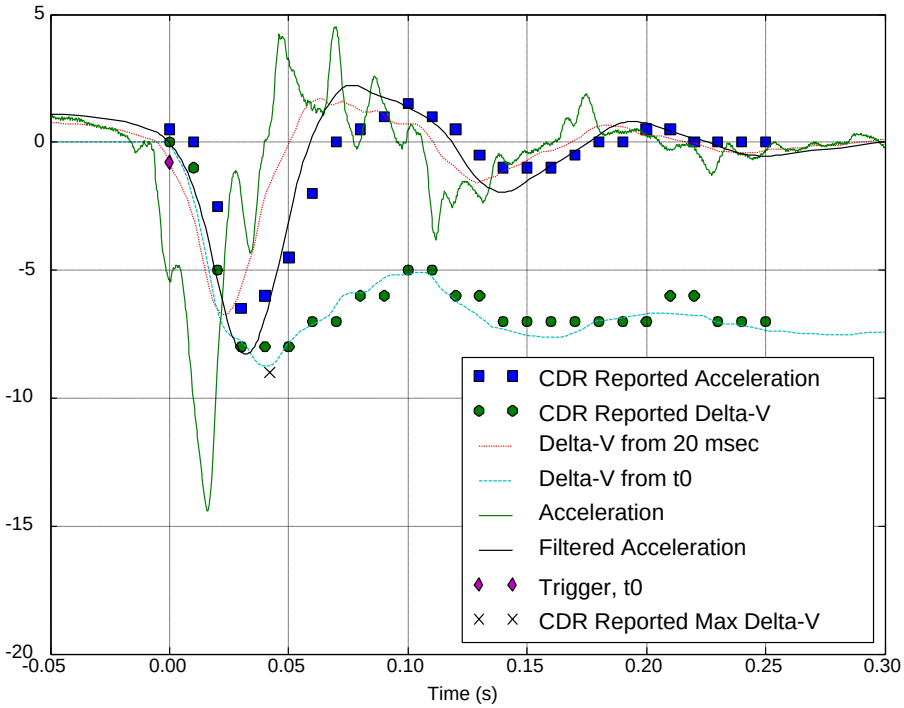


Figure 52: Crash data from the Civic corresponding to Figure 51.

3.2.2.5 Timing Between 0 and -0.5 Data Points in the Pre-Crash Data

Because the Honda pre-crash data points are labeled in 0.5 second increments, from -5.0 to 0.0, it was originally assumed the spacing between data points was uniform and plotted the data that way accordingly. However, after synchronizing the "0.0" data point precisely at t_0, in several cases the -0.5 and earlier data points appeared to change values before the CAN bus values did (see Figure 49, the red square at "-2"). This is not possible; the EDR cannot anticipate changes in the CAN bus data, it can only report them after the fact. After extensive analysis, the authors developed a working theory that the data point labeled "0.0" is taken at or near algorithm wake up, and can be anywhere between 0 and 0.5 seconds after the data point labeled "-0.5". This concept comes from observing Toyota EDR's which take one last data point at algorithm enable. Toyota resets a timer after each regular-interval data point is written and reports the interval from the next to last data point to AE. The Honda data limitations as of this writing offer no information about last data point timing relative to the "-0.5" point.

EDR files from six 113 km/h (70 mph) hard braking runs were examined. After synchronizing the "0.0" data point, both speed and RPM channels were examined and the last 10 data points were shifted right to eliminate any "anticipation" by the EDR data ahead of the CAN bus data. The revised time from "0.0" to "0.5" was 0.15, 0.15, 0.15, 0.2, 0.25, and 0.35 seconds. Other runs may need only a slight shift to 0.40 or 0.45 seconds and the need for the small shift is not that apparent.

3.2.2.6 Dynamic Steering Maneuvers

The CDR steering data of the 2012 CR-V was assessed via replay of dynamic steering CAN messages shown in Figure 53. This graph also shows the CAN indicated speed (0x309), speeds retrieved from the SRS, and the external accelerometer pulse showing the location of Algorithm Enable (t_0). The non-deployment event was programmed to fire at the same time in the CAN history for 5 runs. From Figure 53, it appears that the SRS steering data tracks the CAN data closely for aggressive steering inputs. However, the scale on Figure 53 is such that differences between CAN steering data and CDR reported data are difficult to detect. Therefore, the difference between the data retrieved from the SRS and the CAN data are calculated for each run and plotted in Figure 54. The data at time = -4.5 in Figure 54 shows a possibility of a spread of 15 degrees for the case of transient steering maneuvers. Most other data points retrieved from the SRS are within 5 degrees of the CAN data.

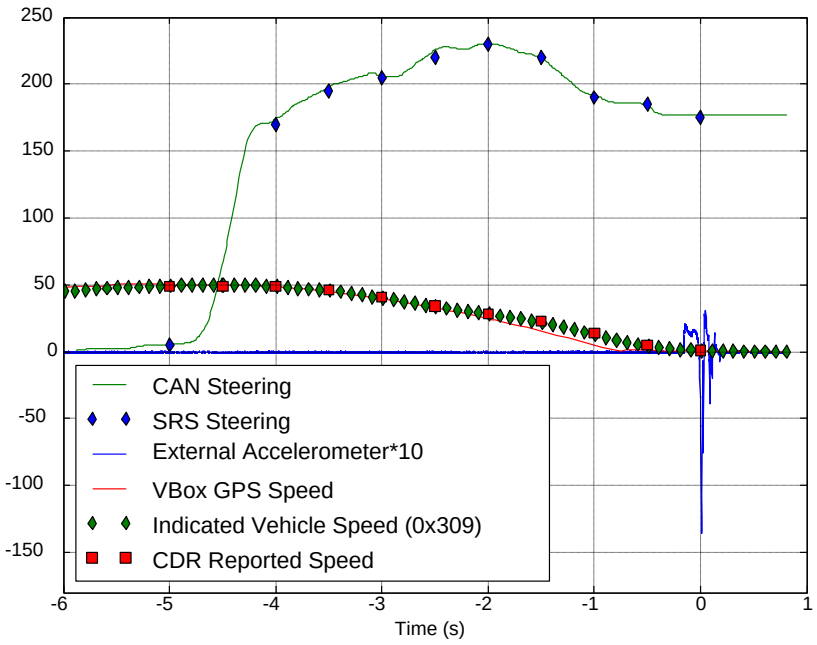


Figure 53: CR-V Pre-Crash Data Dynamic Steering Maneuver

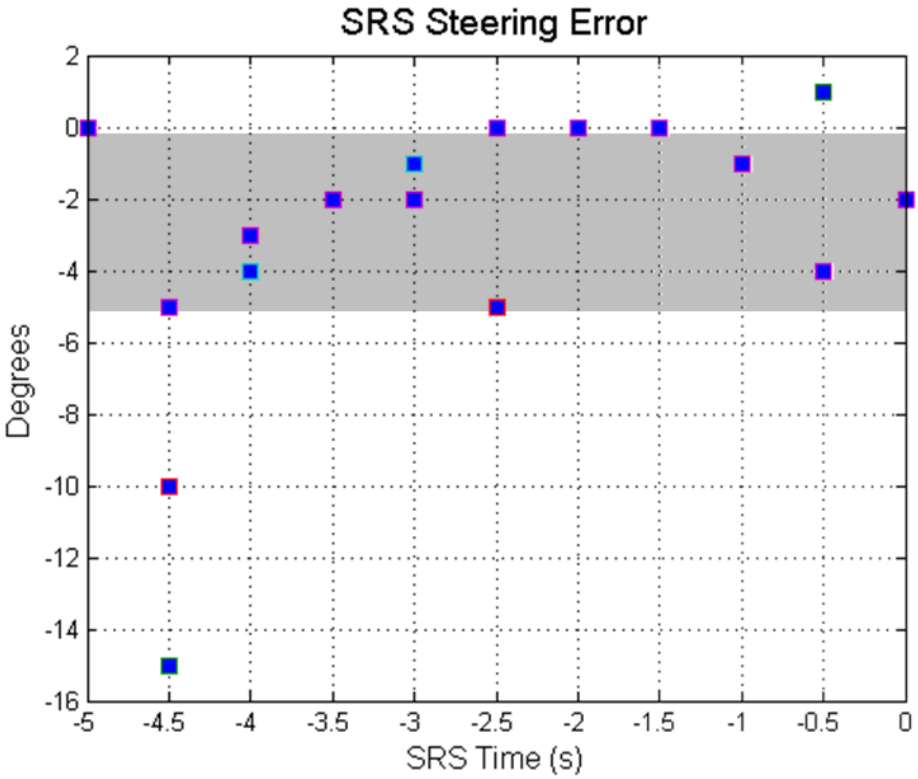


Figure 54: SRS steering data minus the CAN data for the CR-V.

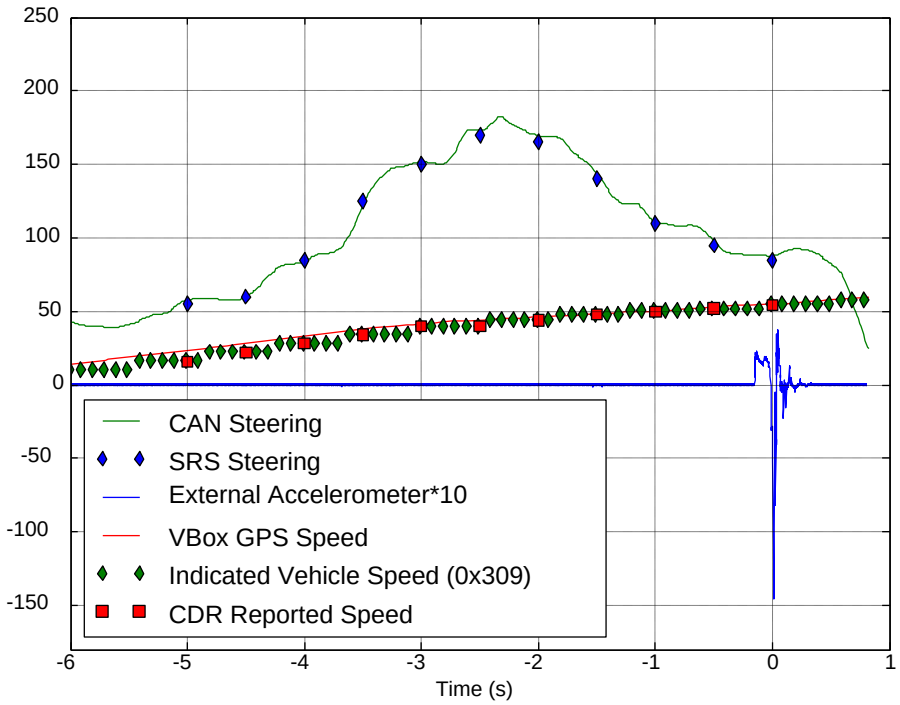


Figure 55: Civic Pre-Crash Data from a dynamic steering maneuver.

Similarly, the Civic SRS steering data was assessed through the CAN replay of a dynamic steering CAN messages which are shown in Figure 55. There are two significant sources for differences in the steering signal. The first is a value or truncation error that manifests itself as a vertical difference on a time history plot like the ones shown in Figure 54. The second error source is from a recording delay or difference along the horizontal (time) axis. The important observation from these tests is that the steering appears to truncate the CAN value and the SRS data provides a rough estimate of steering input.

3.2.3 Passenger Car EDR testing based on Simulation

Through use of simulation output we aim to generate CAN messages specific to the 2012 Honda Vehicles used in this study. The methodology used to achieve this is summarized in Figure 56 as a flow chart. HVE is a physics simulation program sold by Engineering Dynamics Corporation for modeling vehicle dynamics and traffic crashes.

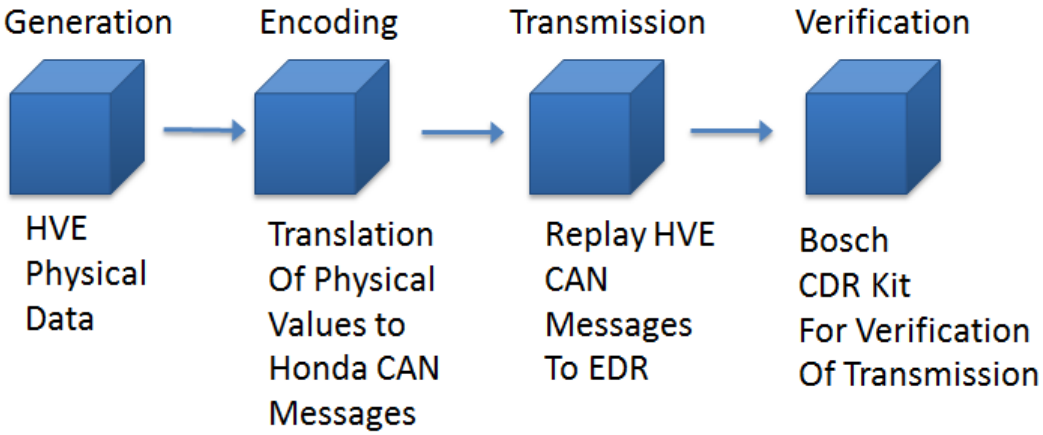


Figure 56: HVE to CAN Transcription Overview

3.2.3.1 An Introduction to HVE

NHTSA sponsored a project in the early 1970's to develop a uniform and accurate program to interpret physical crash data, from which McHenry published Simulation Model of Automobile Collisions (SMAC) in 1971 [45]. This program operates in a two dimensional environment with 3 degrees of freedom (x, y, and yaw). The development of SMAC was limited in the 70's by the lack of computer memory space and expense. The increase in computer power along with further research of automobile dynamics has allowed simulation models to become more robust over the years. Currently, Engineering Dynamics Corporation (EDC) produces a simulation and reconstruction software package called HVE (Human Vehicle Environment). HVE models vehicle dynamics, simulates damage done during collisions, car trajectories pre and post collision, initial speeds, yaw rates, etc. HVE offers multiple algorithms to produce the simulation outputs, namely EDSMAC4, SIMON, EDCRASH, EDSVS, EDVDS, and EDCRASH4. The SIMON (SImulation MOdel Non-linear) algorithm will be used in this paper. EDC has published multiple papers through SAE validating the accuracy of the HVE SIMON algorithm [46]. SIMON is a 3D physics based algorithm which allows for six degrees of freedom (x, y, z, roll, pitch, and yaw). The SIMON algorithm takes user specified

input values and use a time forward Runge-Kutta integration method to predict simulation outputs.

3.2.3.2 Purpose for Simulations

The EDR testing methodology presented in this paper requires a CAN history for the dynamic event intended to be studied be available (e.g. a maximum ABS braking or high speed dynamic steering). Having such records is not common. For example, one must have the vehicle of interest and appropriate testing equipment to gather such data. Translating HVE output into CAN messages removes the need to gather highly dynamic CAN histories for replay. If both the message location and bit resolution of the EDR CAN data is known, the simulation output may be translated into CAN messages. This transcription process would allow researchers to simulate the event, translate it to CAN messages, replay the event to the EDR module, and compare the simulated EDR history to that of the actual EDR history. This translation process may make the results of an accident reconstruction simulation more convincing by allowing the reconstructionist to account for unknown errors in the transfer functions of the EDR itself. This ability would also aid in the evaluation of EDR data by allowing more potentially dangerous maneuvers to be studied (e.g. 100mph maximum ABS breaking or 70mph dynamic steering tests) giving us a better understanding of the performance of such devices at higher speeds. There are, as will be demonstrated in the proceeding sections, complications that may arise in this process.

3.2.3.3 HVE Simulation

A dynamic steering maneuver was simulated in HVE. In this simulation a SUV traveling with an initial velocity of 34.67 mph was made to steer with a constant 210° input on a flat asphalt surface. The specific user inputs concerning this simulation as well as a pictorial summary of the simulation are given by Figure 57 though Figure 62. This simulation uses a Ford Escape as the test vehicle since the Honda CR-V was not available in the HVE vehicle library and both cars have similar geometries. The weight of the Ford was altered, as shown in Figure 60 to the weight of a stock 2012 Honda CR-V.

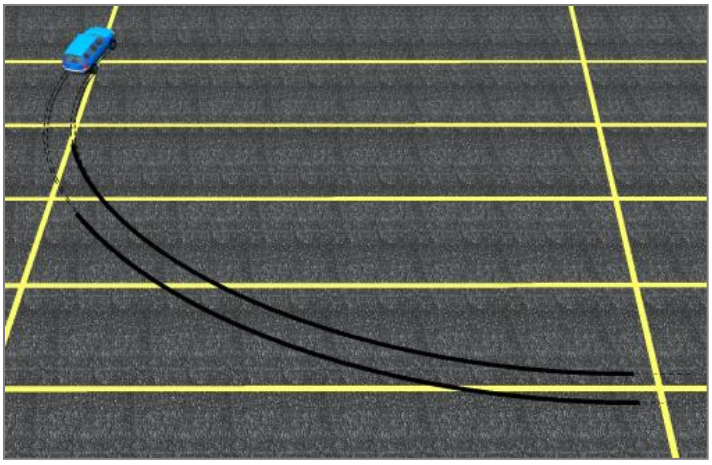


Figure 57: Graphical output from HVE showing a hard steering maneuver.

Figure 58: HVE driver controls showing steering input.

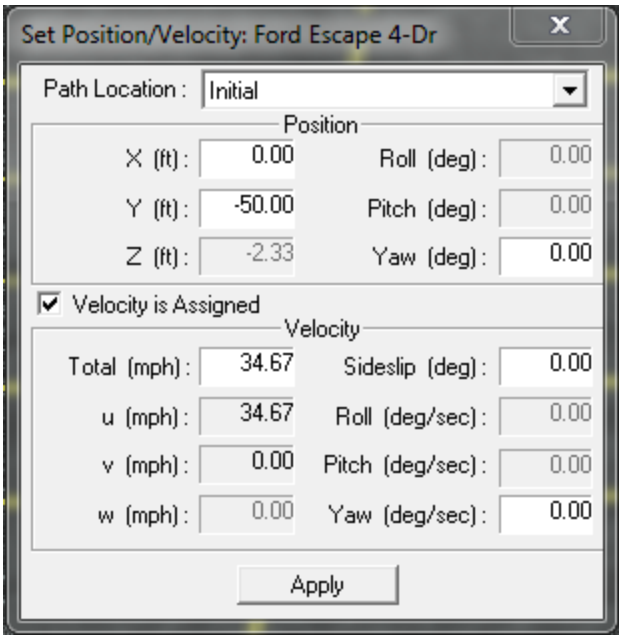


Figure 59: HVE Initial Position/Velocity Inputs

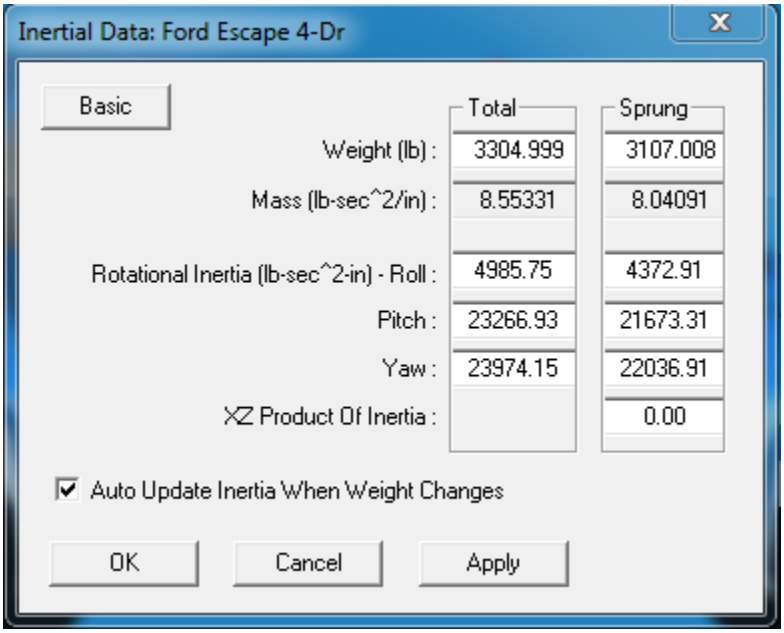


Figure 60: HVE Vehicle Inertial Data

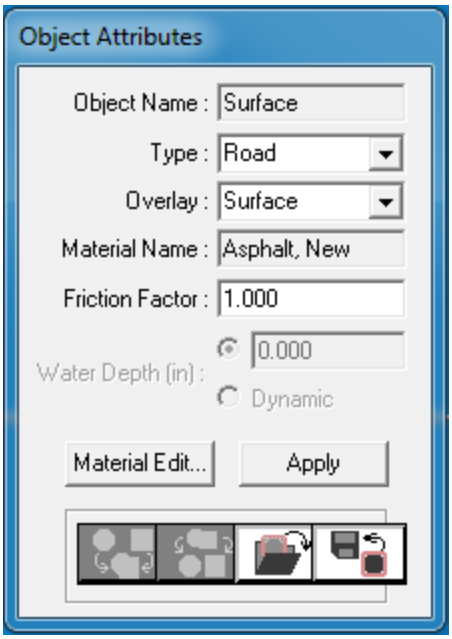


Figure 61: HVE Environment Surface Data

3.2.3.4 HVE Simulation Playback Results

Upon replaying the converted HVE CAN file to the SRS module an unexpected result was found. There appears to be a checksum which validates the data accepted by the SRS module and if that checksum is not correct the SRS module will hold the previously accepted value. In this Figure, the solid line which has the values of 0 and 210 corresponds to transmitted CAN steering, the blue diamonds, which have values of only 0 and 210, represent the SRS reported steering, the green diamonds represent the CAN speed, the red squares represent the CDR reported speed, and the acceleration pulse of the apparatus is marked by the blue spike at approximately t=0. The test corresponding to Figure 62 was done using the CR-V SRS module. As previously shown, the CR-V module both refreshes and updates speed values every 0.1 s. However, this record clearly shows a large delay in the updating of the speed value (approximately 1.5s). To ensure that the testing apparatus and python translation script were functioning properly, the transmitted HVE CAN data was logged during a test. This test showed that the transmitted HVE CAN messages were appropriately transferred in regards to message timing and message value. Upon further inspection it was found that if byte 7 of CAN ID 0x309 is removed from the CAN record no speed values will be updated to the SRS module (remember that only bytes 4 and 5 of 0x309 are responsible for the speed value). These test have led to the conclusion that byte 7 of 0x309 may function as a checksum. It was attempted to discover the checksum method without success. This checksum is believed to be present for all SRS CAN sources as the steering was also not updated for 2.5s, which is much longer than its 0.01s refresh rate.

The discovery of the checksum makes falsifying EDR records much more difficult and adds a layer of security to the validity of the reported data.

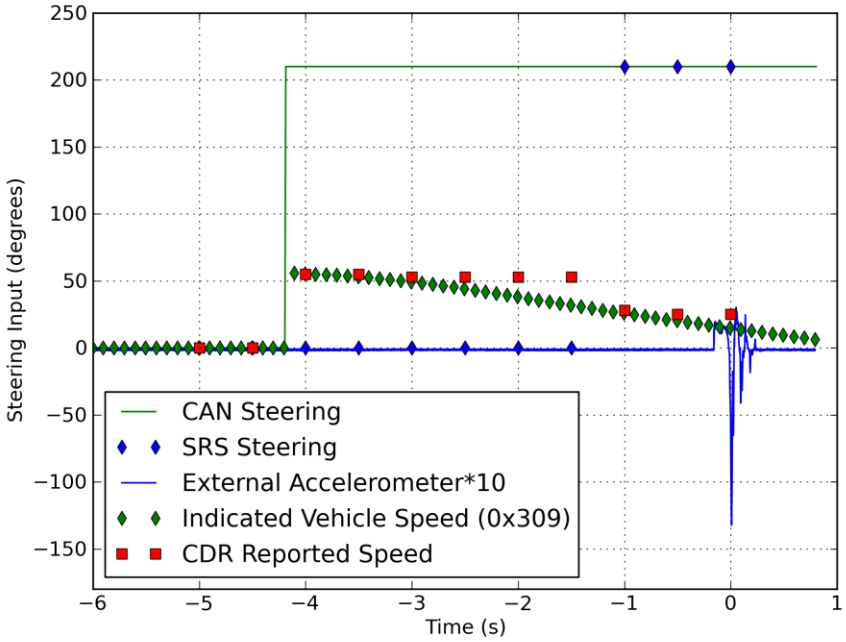


Figure 62: HVE CAN SRS Playback Results

3.3 Heavy Vehicle EDR Forensic Extraction Results

3.3.1 Forensically sound data extraction from a Caterpillar ECM

Our proposed solution was implemented and validated by performing a data extraction and replay test on a Caterpillar ECM. The underlying design requirements were written in such a way that our solution could be easily used for other manufacturers.

The information stored on the CAT ECMs under study include warranty report information and snapshot information. The warranty report contains the identity of the ECM, historical usage information such as engine use histogram, logged fault codes, and engine configuration information.

"Snapshots" are a freeze-frame of the state of the truck at the time a critical fault is detected. These snapshots include everything from wheel speed to engine speed. As a bench download can lead to new snapshots being created, overwriting existing snapshot information, logging and replaying snapshot information is critical to any forensic solution for Caterpillar ECMs.

Observation of RP1210 calls made by the Caterpillar ET software showed that all requests sent by, and all responses to those requests, were made using extensions to the J1708/J1587 and J1939 protocols proprietary to Caterpillar. As requests and responses changed significantly between extractions, with no change in data displayed, it was determined that some session-based encryption mechanism was used.

The names of the functions obviously suggest that much of the message traffic is encrypted. Analysis revealed that Caterpillar follows these steps:

1. CAT ET sends a session key to the ECM using a SecuritySetup message.
2. The ECM sends a session key to CAT ET using a SecuritySetup message.
3. For each encrypted message, an individual key is generated by summing the current session key and the second nibble of the proprietary PID.
4. Each message is passed to a native DLL along with its key for decryption.
5. The key is an index into an array of bytes; the relevant byte is XOR'd with each byte in the message to encrypt/decrypt it.

With the algorithm that encrypted ECM communications was known, the information contained in those messages could be observed to determine how that information should be extracted and replayed. The API hooking tool was extended to decrypt ECM communications on-the-fly and log them when they were intercepted. It was discovered that the protocols followed a specific pattern (see Figure 63: Example exchange in the CAT ATA Protocol).

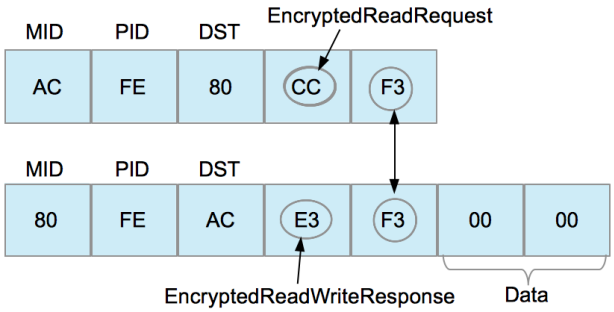


Figure 63: Example exchange in the CAT ATA Protocol

Armed with this information, it was now possible to extract the information. A manual extraction was performed according to a checklist for a crash information extraction, and the RP1210 API calls were logged.

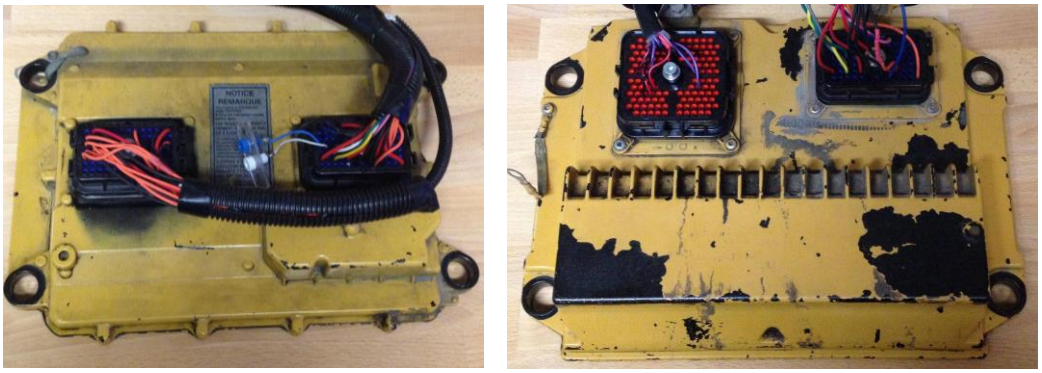


Figure 64: Two CAT ECMs used for testing.

Each request was logged in a plaintext format as it was sent. It should be noted that the storage format preserves the opcode of the message. After the message was sent, proprietary responses with matching PIDs were recorded in the key-value pair. It was observed that some responses were split over several messages, so this was accounted for in the extraction software.

In order to ensure that the method of information extraction was reliable, replays from the ECM were tested against actual extractions from CAT ECMs. The ECMs tested were evidence ECMs that were already used, and thus had been pre-populated with data. The procedure was used for extraction and replaying in two CAT ECMs (Figure 64: Two CAT ECMs used for testing).

Using the following procedure, we were able to validate our solution:

1. Perform one extraction of test ECM according to checklist while recording API calls. Save Warranty Report information and record snapshot information.
2. Repeat step one twice more, each time saving Warranty Reports and recording snapshots.
3. Extract ECM data using logged RP1210 calls obtained in step 1.

4. Perform 3 replayed extractions, using data extracted in step 3.
5. Compare ECM extractions to find differences in data contained therein.
6. Compare replayed extractions to find differences in data.
7. Compare consistency of ECM extractions and replayed extractions.

In addition, snapshot data can be recorded manually and comparisons were then carried out by hand. Warranty report information, however, is stored in a plain-text XML format. Accordingly, the data contained in these files was compared using a suite of tools developed by Adrian Mouat [47] for XML comparison. Tests were also conducted to ensure that our cryptographic protection mechanism detects any changes made to extracted data.

4 Conclusions

4.1 Conclusions Regarding Cyber Physical System Analysis

4.1.1 CAN Logging Design and Analysis

ACL #2 and ACL #3 adequately captured CAN traffic, preserving order and content, while satisfying temporal fidelity within acceptable ranges of tolerance. The inclusion of an OpenLog Chip in both solutions appeared to play a factor in freeing the Arduino main processor from logging tasks, allowing it to dedicate resources to perform data acquisition and transceiver tasks. These solutions represent inexpensive and practical alternatives for logging CAN traffic on vehicles.

4.1.2 Real time Replay Methodology: Logger Evaluation

Of the logger solutions evaluated using the Real time replay methodology, only ACL #3 (the Arduino Due solution) performed adequately, and only after a modification that replaced the serial interface with a built-in USB port. Still, considering that the Arduino Due solution costs under $100, it is an ideal solution for many CAN research projects. If necessary, optimizing the presented Arduino solutions, e.g. binary data transmission, should help to improve the Arduino Due solution even more and make it possible to handle CAN transmission speeds over 500 kB/s.

4.1.3 Real-time Replay Methodology: System Characterization

System characterization of an internal voltage sensor for a Transmission Control Unit demonstrated the utility and fidelity of the real-time replay methodology and framework. Testing the TCU voltage sensor revealed a systematic underreporting of voltage by the sensor. It was further determined that the inferred conversion delay does not have a significant impact on observed measurements. More importantly, the testing process and methodology is readily applied to other sensors and CAN components. Finally, results from the TCU were compared against a simulated TCU, demonstrating the potential role of simulation in CAN experimentation under real time replay.

4.1.4 Formal Verification Study

Experimentation with Keymaera and hybrid programs exposed fundamental issues with applied formal methods for hybrid systems. Any system must be fully describable with the specification to be suitable for formal analysis. The idiosyncrasies of the system under analysis must be completely captured within the mathematical definitions of the model specification or they will not be considered.

This is not a shortcoming of Keymaera but rather a framing requirement of the underlying formal model theory and is acceptable when studying or designing well-defined systems. The proprietary nature of the TCU voltage system characterized and analyzed above, as well as the use of the Controller Area Network technology as a communication infrastructure formal modeling and analysis remains an extreme challenge.

4.1.5 Implications for policy and practice:

The analytical framework presented offers a collection of tools and processes by which to conduct practical study of vehicle CPSs operating over a CAN infrastructure. Inexpensive alternatives for CAN logging may create new opportunities for technology innovation. The real time replay methodology may encourage more validation and greater system assurance as it offers new economies of scale for testing. It also prescribes an integrated view of empirical analysis, simulation and formal verification.

4.2 Passenger Vehicle Data Accuracy and Testing

There are two major contributions of this line of research in this report: 1) a new methodology to non-destructively and repeatedly test the accuracy of different pre-crash data elements in an event data recorder and 2) applying those techniques to two 2012 Honda vehicles.

The new methodology eliminates the risk of accidentally deploying airbags while gathering GPS and CAN bus data in the test vehicle. The techniques presented in this paper allows gathering of data in vehicle without tampering with the airbag control module. The new methodology allows for repeatable testing and mapping the transfer functions between the vehicle CAN bus data and the EDR. Should a manufacturer make a design change to an air bag based EDR, identical inputs can be given to exemplar control modules from before and after the changes to document any change in the transfer function. This methodology allows researchers the ability to recreate events of interest in a low-cost, repeatable manner.

4.2.1 2012 CR-V Speed Data

Under normal driving conditions that included moderate acceleration and braking, the 2012 Honda CR-V vehicle speed CAN bus message (speed, vehicle indicated) accurately represented the vehicle ground speed. The difference between the VBOX GPS speed and the CAN bus speed was not dependent on vehicle speed, which indicates that the vehicle was properly calibrated. The EDR truncated the speed to the next lower whole km/h. Recalling that the sign convention used was Error = GPS

speed – EDR speed, the truncation increased the average difference by approximately 0.5km/h. The resulting EDR to VBOX differences were between +/- 1.15 km/h with a mean of +0.033 km/h and standard deviation of 0.39 km/h. This suggests that the CR-V normal driving speed data is accurate to about 1% at speeds near 100 km/h as tested with new, minimally worn tires.

Under dynamic hard braking conditions, as expected, the wheel speeds under report the GPS ground speed due to wheel slip. The CAN bus vehicle indicated speed data updated approximately every 0.1 seconds, but under hard braking conditions the reporting lags the ground speed. This reporting delay results in reporting an earlier, higher speed than the current actual speed by up to 10km/h, and more than offsets the under reporting effects of wheel slip.

4.2.2 2012 Civic Speed Data

Under steady state conditions the 2012 Honda CR-V vehicle speed CAN bus message (speed, vehicle indicated) accurately represented the vehicle ground speed. The difference between the VBOX GPS speed and the CAN bus speed was not dependent on vehicle speed, which indicates that the vehicle was properly calibrated. The EDR truncated the CAN speed to the next lower whole km/h, resulting in the average GPS-EDR difference being higher by approximately 0.5km/h, to a mean of +0.22 km/h with a standard deviation of 0.33 km/h. The range was from -1.38 km/h to +1.95 km/h. This corresponds to accuracy within about 2% at speeds near 100km/h as tested with new, minimally worn tires.

Under dynamic hard braking conditions, as expected, the wheel speed under-reported GPS measured ground speed due to wheel slip. The CAN bus vehicle indicated speed lagged the true ground speed. While a CAN bus vehicle speed message was transmitted every 0.1 seconds, under some circumstances the value only updated every 0.6 seconds. This significant reporting delay results in reporting an earlier, higher speed than the current actual speed, by up to 20km/h, which more than offsets the under reporting effects of wheel slip.

4.2.3 Other SRS Reported Data

The steering angle recorded in the SRS module is truncated with a resolution of 5 degrees. For negative steering angles, the truncation is towards zero. No anomalies were observed in other parameters such as accelerator pedal position, brake on/off, or engine speed.

4.3 Conclusions Regarding Heavy Vehicle EDR Forensics

One of the goals of the project was the development of a methodology for forensically sound extraction of ECM data. While the field of digital forensics has well defined and accepted methods in the IT world, mostly through the imaging and protection of hard drives, the world of digital forensics in heavy vehicle systems is still in its infancy and it not nearly as strong and robust as it should be. We have also illustrated the weaknesses associated with current practices and developed a new methodology that could have a significant impact on this domain.

Our verification results show that using our solution, the original evidence is modified less by using forensic replay than it is by repeatedly extracting the information by traditional means. Even if no additional faults are created during a bench download, the ECM running time is better preserved by imaging and replaying the ECM data rather than repeated downloads.

The expertise requirement is not totally alleviated by the extraction and replay process; an investigator still has to know how to connect to the ECM and power it up properly. However, the main advantage of the forensic extraction and replay process is that all information extraction is automated; no knowledge of diagnostic software or the steps required to gather pertinent crash information is needed. This is an advantage in a law enforcement context where training time is at a premium.

A solution was also implemented, demonstrated and validated by performing tests on two CAT ECMs. By default, the CAT ECM extraction process is almost completely opaque. Unless certain data are not available on the ECM, or an error occurs during the download process, there is no record kept of the traffic other than its final interpretation by the maintenance software. Our solution addresses that problem by using a forensic replay method that records network traffic. This traffic can be examined after the fact to verify the extraction and replay process.

5 Bibliography

[1] R. Bortolin, S. v. Nooten, M. Scodeller and D. e. al., "Validating Speed Data from Cummins Engine Sudden Deceleration Data Reports," *SAE International Journal of Passenger Cars - Mechanical Systems,* vol. 2, pp. 970-982, 2009.

[2] T. Henzinger, "The theory of hybrid automata," in *11th Annual IEEE Symposium on Logic in Computer Science*, Washington, DC, 1996.

[3] T. A. Henzinger, P.-H. Ho and H. Wong-Toi, "Hytech: A model checker for hybrid systems," *Software Tools for Technology Transfer,* vol. 1, pp. 460-463, 1997.

[4] A. Platzer, "Keymaera: A hybrid theorem prover for hybrid systems," 2013. [Online]. Available: http://symbolaris.com/info/KeYmaera.html. [Accessed 5 July 2014].

[5] NHTSA, "49 CFR Part 563, Event Data Recorders, Final Rule," [Online]. Available: http://www.nhtsa.gov/DOT/NHTSA/Rulemaking/Rules/Associated%20Files/EDRFinalRule_Aug2006.pdf. [Accessed 28 Sept 2014].

[6] A. Diacon, J. Daily, R. Ruth and C. and Mueller, "Accuracy and Characteristics of 2012 Honda Event Data Recorders from Real-Time Replay of Controller Area

Network (CAN) Traffic," *SAE Int. J. Trans. Safety,* vol. 1, no. 2, pp. 399-419, 2013.

[7] Robert Bosch, GmbH, "Controller Area Network (CAN) Specfication, Version 2.0," 1991. [Online]. Available: http://www.bosch-semiconductors.de/media/pdf_1/canliteratur/can2spec.pdf. [Accessed 29 Sept 2014].

[8] C. Mueller and J. P. M. Daily, "Assessing the Accuracy of Vehicle Event Data Based on CAN Messages," in *SAE Technical Paper 2012-01-1000*, doi:10.4271/2012-01-1000, 2012.

[9] A. Chidester, J. Hinch, T. Mercer and K. Schutz, "Recording automotive crash event data," in *International Symposium on Transportation Recorders*, Arlington, VA, 1999.

[10] J. Lawrence, C. Wilkinson, B. Heinrichs and G. and Siegmund, "The Accuracy of Pre-Crash Speed Captured by Event Data Recorders," in *SAE Technical Paper 2003-01-0889*, doi:10.4271/2003-01-0889, 2003.

[11] P. Niehoff, H. Gabler, J. Brophy, A. Chidester, J. Hinch and C. Ragland, "Evaluation of Event Data Recorders in Full Systems Crash Tests," in *19th International Technical Conference on the Enhances Safety of Vehicles*, Washington, DC, 2005.

[12] C. Wilkinson, J. Lawrence, B. Heinrichs and D. and King, "The Timing of Pre-Crash Data Recorded in General Motors Sensing and Diagnostic Modules," in *SAE Technical Paper 2006-01-1397*, doi:10.4271/2006-01-1397, 2006.

[13] C. Bare, B. Everest, D. Floyd and D. and Nunan, "Analysis of Pre-Crash Data Transferred over the Serial Data Bus and Utilized by the SDM-DS Module," *SAE Int. J. Passeng. Cars – Mech. Syst.,* vol. 4, no. 1, pp. 648-664, 2011.

[14] H. Gabler, C. Thor and J. Hinch, "Preliminary Evaluation of Advanced Air Bag Field Performance Using Event Data Recorders," DOT HS 811 015, 2008.

[15] R. Ruth, O. West, J. Engle and T. and Reust, "Accuracy of Powertrain Control Module (PCM) Event Data Recorders," in *SAE Technical Paper 2008-01-0162*, doi:10.4271/2008-01-0162, 2008.

[16] R. Ruth, O. West and H. and Nasrallah, "Accuracy of Selected 2008 Ford Restraint Control Module Event Data Recorders," *SAE Int. J. Passeng. Cars - Mech. Syst.,* vol. 2, no. 1, pp. 991-1001, 2009.

[17] N. Takubo, H. Ishikawa, K. Kato and T. Okuno, "Study on Characteristics of Event Data Recorders in Japan," in *SAE Technical Paper 2009-01-0883*,

doi:10.4271/2009-01-0883, 2009.

[18] N. Takubo, T. Hiromitsu, K. Kato and K. Hagita, "Study on Characteristics of Event Data Recorders in Japan; Analysis of J-NCAP and Thirteen Crash Tests," *SAE Int. J. Passeng. Cars – Mech. Syst.,* vol. 4, no. 1, pp. 665-676, 2011.

[19] R. Ruth and T. Reust, "Accuracy of Selected 2008 Chrysler Airbag Control Module Event Data Recorders," *SAE Int. J. Passeng. Cars - Mech. Syst.,* vol. 2, no. 1, pp. 983-990, 2009.

[20] R. Bortolin, B. Gilbert, J. Gervais and J. and Hrycay, "Chrysler Airbag Control Module (ACM) Data Reliability," *SAE Int. J. Passeng. Cars - Mech. Syst.,* vol. 3, no. 1, pp. 653-674, 2010.

[21] R. Ruth and T. Brown, "2009 Crown Victoria PCM EDR Accuracy in Steady State and ABS Braking Conditions," in *SAE Technical Paper 2010-01-1000*, doi:10.4271/2010-01-1000, 2010.

[22] National Highway Traffic Safety Administration, "Event Data Recorder - Pre Crash Sata Validation of Toyota Products," VRTC Report DCD9157-1WDC, 2011.

[23] R. Ruth, W. Bartlett and J. and Daily, "Accuracy of Event Data in the 2010 and 2011 Toyota Camry During Steady State and Braking Conditions," *SAE Int. J. Passeng. Cars - Electron. Electr. Syst.,* vol. 5, no. 1, pp. 358-372, 2012.

[24] National Center for Statistics and Analysis, "Traffic Safety Facts 2008 Data," National Highway Traffic Safety Administration, 2009.

[25] T. Austin and M. Farrell, "An Examination of Snapshot Data in Caterpillar Electronic Control Modules," *SAE International Journal of Passenger Cars - Mechanical Systems,* vol. 1, p. 1, 2011.

[26] "Serial Data Communications Between Microcomputer Systems in Heavy Duty Vehicle Applications J1708".

[27] SAE International, "Electronic Data Interchange Between Microcomputer Systems in Heavy-Duty Vehicle Applications," Society of Automotive Engineers.

[28] SAE International, "Vehicle Application Layer J1939-71".

[29] "Windows Communication API TMC RP1210".

[30] *Federal Rules of Evidence Rule 702: Testimony By Expert Witnesses.*

[31] M. Meyers and M. Rogers, "Meeting the Challenges of Scientific Evidence," 2005.

[32] NIST, "Disk Imaging Tool Specification," 2001.

[33] N. S. Report, "Electronic Crime Scene Investigation: A Guide for First Responders," 2008.

[34] International Standards Organization, "Guidelines for Best Practice in the Forensic Examination of Digital Technology," 2002.

[35] R. McKemmish, "When Is Digital Evidence Forensically Sound?," *Advances in Digital Forensics,* vol. 4, pp. 3-15, 2008.

[36] E. Casey, "Error, Uncertainty, and Loss in Digital Evidence," *International Journal of Digital Evidence,* vol. 1, p. 1, 2002.

[37] J. Johnson, A. Kongs and J. Daily, "On The Digital Forensics of Heavy Truck Electronic Control Modules," 2014.

[38] T. Reust, J. Morgan and P. Smith, "Method to Determine Vehicle Speed During ABS Brake Events Using Heavy Vehicle Event Data Recorder Speed," *SAE International Journal of Passenger Cars - Mechanical Systems,* vol. 3, pp. 644-652, 2010.

[39] J. Steiner, T. Cheek and S. Hinkson, "Data Sources and Analysis of a Heavy Vehicle Event Data Recorder - V-MAC III," *SAE International Journal of Commercial Vehicles,* vol. 6, pp. 209-228, 2013.

[40] J. Berdajs and Z. Bosnic, "Extending Applications using an advanced approach to DLL injection and API hooking," *Software: Practice and Experience,* vol. 40, pp. 567-584, 2010.

[41] BeagleBoard.org Foundation, "BeagleBone Black," [Online]. Available: http://beagleboard.org/black. [Accessed 28 Sept 2014].

[42] A. Biryukov, D. Khovratovich and I. Nikolic, "Distinguisher and Related-Key Attack on the Full AES-256," 2009.

[4 S. Manuel, "Classification and generation of disturbance vectors for collision

3] attacks against SHA-1," *Designs, Codes and Cryptography,* vol. 59, no. 1-3, pp. 247-263, 2011.

[44] M. Bellare and P. Rogaway, "Optimal Asymmetric Encryption -- How to encrypt with RSA - Eurocrypt '94 Proceedings," 1995.

[45] R. McHenry, "Development of a Computer Program to Aid in the Investigation of Highway Accidents," Calspan Report V3-2979-V-1, Buffalo, NY, 1971.

[46] T. Day, "Validation of the SIMON Model for Vehicle Handling and Collision Simulation - Comparison of Results with Experiments and Other Models," in *SAE Technical Paper 2004-01-1207*, doi:10.4271/2004-01-1207, 2004.

[47] A. Mouat, "XML Diff and Patch Utilities," 2002.

[48] R. Ruth and J. Daily, "Accuracy of Event Data Recorder in 2010 Ford Flex During Steady State and Braking Conditions," *SAE Int. J. Passeng. Cars – Mech. Syst.,* vol. 4, no. 1, pp. 677-699, 2011.

[49] B. Schneier, Applied Cryptography, x, Ed., Joh Wiley \& Sons, 1996.

[50] T. Reust, "The Accuracy of Speed Captured by Commercial Vehicle Event Data Recorders," 2004.

[51] S. v. Nooten and J. Hrycay, "The Application and Reliability of Commercial Vehicle Event Data Recorders for Accident Investigation and Analysis," 2005.

[52] NIST, "Hardware Write Blocker Device (HWB) Specification," 2004.

[53] C. Miller and C. Valasek, "Adventures in Automotive Networks and Control Units," 2013.

[54] J. Lyle, "A Strategy For Testing Hardware Write Block Devices," *Digital Investigation,* vol. 3, pp. 3-9, 2006.

[55] U. Larson and D. Nilsson, "Securing Vehicles Against Cyber Attacks," 2008.

[56] K. Koscher, A. Czeskis, F. Roesner, S. Patel, T. Kohno, S. Checkoway, D. McCoy, B. Kantor, D. Anderson, H. Shacham and S. Savage, "Experimental Security Analysis of a Modern Automobile," 2010.

[57] P. Koopman and T. Chakravarty, "Cyclic Redundancy Code (CRC) Polynomial Selection For Embedded Networks," 2004.

[58] T. Kohno, "Attacking and Repairing the WinZip Encryption Scheme," 2011.

[59] Z. Gutterman, B. Pinkas and T. Reinman, "Analysis of the Linux Random Number Generator," 2006.

[60] J. Day and H. Zimmerman, "The OSI reference model," *Proceedings of the IEEE,* vol. 71, pp. 1334-1340, 1983.

[61] R. Cramer and V. Shoup, "Design and Analysis of Practical Public-Key Encryption Schemes Secure Against Adaptive Chosen Ciphertext Attack," *SIAM Journal on Computing,* vol. 33, pp. 167-226, 2004.

[62] U. S. Court, *Daubert v. Merrell Dow Pharmaceuticals,* 1993.

[63] 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," *Usenix Security,* vol. 1, p. 1, 2011.

[64] N. N. Center and Analysis, "Traffic Safety Facts 2008 Data," 2008.

[65] B. Carrier, File System Forensic Analysis, x, Ed., Addison Wesley, 2005.

[66] B. Carrier, "Defining Forensic Examination and Analysis Tools Using Abstraction Layers," *International Journal of Digital Evidence,* vol. 1, p. 1, 2003.

[67] M. Breeuwsma, M. deJongh, C. Klaver, R. v. der and M. Roeloffs, "Forensic Data Recovery From Flash Memory," *Small Scale Digital Device Forensics Journal,* vol. 1, p. 1, 2007.

[68] E. B. Barker and J. M. Kelsey, "SP 800-90A. Recommendation for Random Number Generation Using Deterministic Random Bit Generators," National Institute of Standards \& Technology, 2012.

[69] P. Amini, *PyDbg: A pure Python win32 debugging abstraction class.*

[7 E. A. Lee, "Cyber physical Systems: Design challenges," University of California,

0] Berkeley, 2008.

[71] R. Ruth and J. Daily, "Accuracy and Timing of 2013 Ford Flex Event Data Recorders," in *SAE Technical Paper 2014-01-0504*, 2014.

[72] T. Austin and W. Messerschmidt, "Electronic Control Module Quick Reference Guide," 2012.

[73] SAE International, "Heavy Vehicle Event Data Recorder (HVEDR) Standard - Tier 1," 2008.

[74] F. Goichon, C. Lauradoux, G. Salagnac and T. Vuillemin, "Entropy transfers in the Linux Random Number Generator," 2012.

[75] W. Messerschmidt, T. Austin, P. Smith, T. Cheek and e. al., "Simulating the Effect of Collision-Related Power Loss on the Event Data Recorders of Heavy Trucks," in *SAE Technical Paper 2010-01-1004*, doi:10.4271/2010-01-1004, 2010.

[76] W. Messerschmidt, "DDEC Reports Version 8.02: Analysis of Daily Engine Usage Data," in *Illinois Association of Technical Accident Investigators*, Peoria, IL, 2013.

[77] D. Plant, T. Cheek, T. Austin and e. al., "Timing and Synchronization of the Event Data Recorded by the Electronic Control Modules of Commerical Motor Vehicles - DDEC V," *SAE International Journal of Commercial Vehicles,* vol. 6, pp. 209-228, 2013.

[78] K. Drew, S. van Nooten, R. Bortolin and J. Gervais, "The Reliability of Snapshot Data from Caterpillar Engines for Accident Investigation and Analysis," in *SAE Technical Paper 2008-01-2708*, doi:10.4271/2008-01-2708, 2008.

6 Dissemination of Research Findings

Publications and presentations resulting from this award are cited as references [6], [8], [23], [37], and [48]. Additionally, A U.S. Patent application was submitted under application number 61/811,004.

Additionally, some of the contents on http://tucrrc.utulsa.edu were generated using support from this award.