Poster: A Methodology for Semi-Automated CAN Bus Reverse Engineering

Alessio Buscemi[*], Ion Turcanu[*†], German Castignani[*], Romain Crunelle[‡] and Thomas Engel[*]
[*]FSTM, University of Luxembourg [†]Luxembourg Institute of Science and Technology [‡]Xee/Eliocity SAS
{alessio.buscemi,thomas.engel}@uni.lu ion.turcanu@list.lu
german.castignani@ext.uni.lu rcrunelle@xee.com

Abstract— **Semi-automated Controller Area Network (CAN) reverse engineering has been shown to provide decoding accuracy comparable to the manual approach, while reducing the time required to decode signals. However, current approaches are invasive, as they make use of diagnostic messages injected through the On-Board Diagnostics (OBD-II) port and often require a high amount of non-CAN external data. In this work, we present a non-invasive universal methodology for semi-automated CAN bus reverse engineering, which is based on the taxonomy of CAN signals. The data collection is simplified and its time reduced from the current standard of up to an hour to few minutes. A mean recall of around 80 % is obtained.**

I. Introduction

Controller Area Network (CAN) is a master-less message-based protocol dedicated for communication among Electronic Control Units (ECUs) within a vehicle [1]. The payload of each message, or frame, sent by any ECU carries signals, which encapsulate real-time information regarding vehicle functions. Despite being the world's most adopted protocol for in-vehicle communication, CAN does not provide encryption. Nonetheless, to secure the information transiting in their vehicle models, car manufacturers encode the CAN data according to their secret proprietary format. The most common way to disclose this format is through reverse engineering (RE). Traditionally, CAN bus RE is performed by a trained human operator who triggers events in the vehicle and visually inspects the CAN traffic to spot changes in real time.

As decoded CAN data is in high demand among researchers and companies providing automotive after-market solutions, the research is focusing on automating the RE process. Automated CAN bus RE can be divided in two main phases: *tokenization*, which consists of identifying the boundaries of every signal within a frame [2] and, *translation*, whose goal is to disclose the actual format of the signals [3]–[5]. Existing solutions are semi-automated (i.e. manual operations have to be performed at data collection time) and typically optimized by using specific Inertial Measurement Unit (IMU) sensors and often rely on intrusive injection of diagnostic messages [2], [4], [5]. In addition, the required data collection time is usually in the order of hours [4].

In this paper, we present a pipeline to perform semi-automated CAN bus RE methodologically and at an unprecedented speed. The novelties introduced in this work concern the data collection and translation phases, which are driven by an extended categorization of the CAN signals.

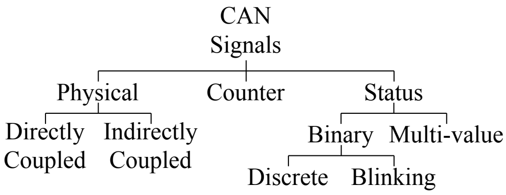

Figure 1. Taxonomy of CAN signals

II. Proposed CAN Bus RE approach

CAN signals are typically divided in three main categories: *physical*, *status*, and *counters* [2], [4], [5]. Physical signals describe the dynamic behavior of the vehicle at driving time. We propose to further divide these signals in two subcategories: *directly coupled* and *indirectly coupled*. Directly coupled signals exhibit a similar behavior among them (e.g. wheel speed) and/or are tied by a clear principle of cause and effect (e.g. the throttle pedal position and the engine RPM). Indirectly coupled signals are those whose correlation with any other physical signal has to be inferred in a non-trivial way (e.g. brake pedal position and engine RPM). Status signals are related to vehicle functions whose values represent a limited set of options. In related work, they are subdivided in two sets, *binary* (on/off) and *multi-value*. We propose to further sub-categorize binary signals into *discrete* and *blinking*. Once triggered, the former maintain their new value (until the status changes again), while the latter periodically change their value from on to off and vice versa. Finally, counter signals display a cyclic behavior. The key to perform an accurate and time-efficient CAN bus RE is to take into consideration the taxonomy of the signals, shown in Figure 1, during the data collection.

Regarding physical signals, it is necessary to collect a CAN log, or *trace*, for each group of directly and indirectly coupled signals. Specific actions have to be performed to capture the dynamic behavior of the searched signal or group of signals. Then, each trace must be tokenized considering the byte order, or *endianness*. Executing the tokenization on traces from multiple driving sessions likely results in finding slightly different sets of tokens, due to the fact that the most significant bits of some signals might be stimulated more in a scenario rather than in others. We recommend establishing the final set of tokens through a likelihood score that takes into account the nature of the signals addressed in each trace. Physical signals can be *unsigned* or *signed*, based on whether they can

13th IEEE Vehicular Networking Conference (VNC 2021)

Table I
Evaluation physical and counter signals

	T1	T2	T3	T4
Mean Recall	85.0 %	88.8 %	66.7 %	80.0 %
Mean NRMSE	2.1 %	4.9 %	4.5 %	1.0 %

Table II
Evaluation status signals

	T5	T6	T7	T8	T9
Mean Recall	95.8 %	72.8 %	57.1 %	80.0 %	87.5 %

assume only positive or negative values. An efficient way to assess the *signedness* of a signal is to parse its information into raw unsigned decimal values throughout the trace, and spot eventual incoherent behavior. To make sense of the information contained in a signal s, it is necessary to parse its binary format into a raw decimal value r, and apply a scale factor f and an offset o, such that $v_s = f_s \cdot r_s + o_s$. This can be done by correlating the signal with external real-time information providing reference values, e.g. GPS speed.

Similarly to physical signals, one trace has to be collected for an individual status signal or group of them. Also, a trace where no operation is performed, called *reference trace*, provides the ground truth to compare any other trace with and, thus, identify changes in the CAN traffic. Usually, while it is easy to spot these changes, correctly identifying the signals of interest is not straightforward. In fact, a single action in the vehicle can trigger multiple signals other than the ones currently researched. Noise can be preliminarily reduced by discarding signals based on temporal patterns that cannot reflect a human action, e.g. a signal switching on and off in 0.1 s cannot possibly represent a door being opened and closed. The next step consists in distinguishing between discrete and blinking signals. For this task, it is essential to preliminarily identify their default value (the value of the signal when it is not triggered), which can be 0 (inactive) or 1 (active).

Finally, counters can be identified based on the a monotonic growth followed by a drop in their value (reset) that they cyclically display. This behavior enhances both the tokenization (as all their bits are activated) and translation.

III. Performance Evaluation

We have designed a CAN bus RE tool able to decode signals carrying a total of 24 distinct vehicle functions, based on the methodology presented in Section II. It is to be noted that a number of status signals are not present in some vehicles. For instance, high-end cars typically have one dedicated ECU for each seat belt, while some low-end cars have only one for the front-left seat belt. For this reason, we assess the performance based on the signals that can actually be found in each vehicle.

To test our tool, we have collected 10 CAN traces for each of 5 vehicle models from 5 distinct manufacturers. Three of these traces are collected to decode the following physical signals: vehicle/wheels speed (T1), throttle pedal position/engine RPM (T2), and steering wheel angle (T3). One trace is to decode the fuel consumption counter (T4). Six traces are used for status signals: doors (T5), seat belts (T6), indicators (T7), air conditioning (T8), and wipers (T9). One trace is the reference trace. The data collection was conducted using a Raspberry Pi 3, equipped with a Pican 2 Duo interface and synchronized with a Xee Connect[1] providing GPS speed information. For the validation, we use the ground truth provided by a partner company expert in car data telematics and reverse engineering.

Tables I and II present the aggregated performance obtained on all vehicles. The recall corresponds to the number of correctly translated signals over the number of signals to translate. The Normalized Root Mean Squared Error (NRMSE) is the difference between the time series of the physical signal parsed with its reference format (i.e. ground truth factor and offset) and the calculated format (i.e. outputted factor and offset), on a test CAN trace. With an average of 25 s per trace, or 5 min in total, the data required to identify and decode up to 27 vehicle functions is one order of magnitude inferior with respect to current state-of-the-art solutions, while achieving similar performance [4], [5].

IV. Conclusion

We present a methodology to perform fast and scalable semi-automated CAN bus RE, based on the taxonomy of CAN signals. Particular attention is paid to the data collection process, which is decomposed in multiple steps and reflects systematically the semantic of the researched signals. In addition, we offer indications on how to identify and translate signals based on their characteristics and format. We validate this approach by implementing a CAN bus RE tool based on it. Our solution requires significantly less time for data collection compared to other state-of-the-art solutions. Future work includes testing on a wider number of signals and further optimization of the RE process.

Acknowledgement

We acknowledge support from the National Research Fund (FNR) under grant number PRIDE15/10621687. We thank Xee for the provided datasets we used to validate our solution.

References

[1] C. Electronics. “CAN Bus Explained.” (2021), [Online]. Available: https://www.csselectronics.com/screen/page/simple-intro-to-can-bus/.

[2] M. Marchetti and D. Stabili, “READ: Reverse engineering of automotive data frames,” *IEEE Transactions on Information Forensics and Security*, vol. 14, no. 4, pp. 1083–1097, 2018.

[3] A. Buscemi, G. Castignani, T. Engel, and I. Turcanu, “A Data-Driven Minimal Approach for CAN Bus Reverse Engineering,” in *3rd IEEE Connected and Automated Vehicles Symposium (CAVS)*, Victoria, Canada: IEEE, Oct. 2020.

[4] M. D. Pesé, T. Stacer, C. A. Campos, E. Newberry, D. Chen, and K. G. Shin, “LibreCAN: Automated CAN Message Translator,” in *Proceedings of the 2019 ACM SIGSAC Conference on Computer and Communications Security (CCS)*, ACM, 2019, pp. 2283–2300.

[5] M. E. Verma, R. A. Bridges, J. J. Sosnowski, S. C. Hollifield, and M. D. Iannacone, *CAN-D: A Modular Four-Step Pipeline for Comprehensively Decoding Controller Area Network Data*, 2020. arXiv: 2006.05993.

[1] PiCAN 2 Duo: www.skpang.co.uk, Xee Connect: www.xee.com