Automated reverse-engineering of CAN messages using OBD-II and correlation coefficients

Bram Blaauwendraad
University of Amsterdam
bram.blaauwendraad@os3.nl

Vincent Kieberl
University of Amsterdam
vincent.kieberl@os3.nl

***Abstract*—Intrusion Detection Systems (IDSs) for the Controller Area Network (CAN) bus often rely on CAN frame characteristics to detect intrusion, because the contents of CAN frames are not generalizable to different vehicle makes and models. However, the use of data plausibility checks may improve accuracy of such IDSs. Research has been conducted into mapping known On-Board Diagnostics version II (OBD-II) parameters to CAN IDs by searching for one-on-one matches between OBD-II values and values in CAN frames in order to obtain knowledge about what data is stored in which CAN frame. Nonetheless, an additional method is required for vehicles in which OBD-II values do not directly match CAN data. In this research, we propose a method to match CAN data to OBD-II values using the Pearson correlation coefficient. Our results show that we can correctly match CAN data to OBD-II parameters on multiple vehicles with some practical limitations.**

***Keywords*—** CAN, OBD-II, reverse-engineering, Audi, Hyundai, Pearson, Correlation, ELM327, intrusion detection, PID, IDS

I. Introduction

In recent years, more and more components of vehicles that were previously controlled by mechanical systems have been replaced in favor of electronic components. These electronic components in automobiles are referred to as Electronic Control Units (ECUs), which control various subsystems of a vehicle such as engine, drivetrain and transmission. ECUs process signals from actuators and sensors, and relay this information to other relevant ECUs over an automotive network called the Controller Area Network (CAN). CAN is a bus network, which means that frames on the CAN bus are broadcast to all other nodes, and there is no notion of a sender or receiver. Instead, what information is contained in a CAN frame is identified by the CAN ID in the header, i.e., a CAN frame with a specific ID may contain the engine coolant temperature at one specific byte index, and the engine oil temperature at another. However, this mapping is not standardized, meaning that which CAN ID is used for what information is entirely at the discretion of the manufacturer [1].

Seeing that CAN was developed in the 1980s and therefore lacks features for security such as authentication and encryption, research has been conducted into Intrusion Detection Systems (IDSs) for automotive applications. These IDSs often use CAN frame characteristics as a feature, as opposed to the actual content of a CAN frame, because this content is not standardized [2, 3, 4, 5]. If it were possible to use CAN frame content as a feature for IDSs so that, for example, the plausibility of a certain value within that CAN frame can be used as a feature, this might provide new opportunities for more accurate intrusion detection within CAN bus systems in vehicles. For this approach to be fruitful, the meaning of the data within a CAN frame would have to be reverse-engineered for every vehicle model. Considering that this is a time-consuming task, it would be convenient if this can be automated. Kang et al. have conducted research into automating the process of reverse-engineering CAN ID meanings using the On-Board Diagnostics-II (OBD-II) interface that is found on all cars today (see Section II for further information) [6]. This system allows mechanics to request parameters such as current engine RPM, engine coolant temperature, etc. through the OBD-II port located inside the vehicle. Kang et al. have used these known requested values to search for matching values in CAN frames that were sent at approximately the same time, hypothesizing that if a value of a certain byte and CAN ID repeatedly matches the value that was obtained through an OBD-II request, it can be assumed that the requested value is indeed contained at that byte location [6]. However, this method does not hold on all vehicle makes and models, because some vehicle manufacturers do not store the human-readable value in a CAN frame directly. Instead, a translation is used. For example, in an Audi A4 B7 vehicle, to be able to store a temperature ranging from -48 to 143 degrees Celsius, an unsigned 8-bit integer stores the temperature as $0.75 \times A - 48$, where A is the unsigned 8-bit integer stored in the CAN frame. This was documented in [7], and we have verified this source using our test vehicle, which produced results that are in line with the translation formulas.

Considering that Kang et al.'s approach only searches for direct one-on-one matches, values that are translated cannot be found using this method. We therefore propose an additional method for reverse-engineering CAN meanings using OBD-II that uses the Pearson correlation coefficient on two data series to indicate whether a potential match has been found. We use the correlation coefficient because if one data series directly relies on the values of another data series, the correlation coefficient will approximate to 1. We show that using this approach, we can find potential matches for translated values.

II. Background

A. Controller Area Network (CAN)

As briefly noted in Section I, the Controller Area Network (CAN) is a bus network that is often used in vehicles. CAN uses a twisted wire pair that makes up a shared backbone between nodes on the network, providing relatively high data throughput of up to 1 Mbps while providing cost efficiency and easy installation. The CAN frame header does not provide a field for sender or receiver. Instead, all messages are broadcast on the bus and are therefore received by every node on the network, including the sender. Nodes utilize frame filtering to discard the frames that are not of use to them by checking an 11-bit header field called the *CAN identifier* (ID). This CAN ID uniquely identifies a frame type on the network, and is therefore in practice used to convey what data is stored in a message with a specific CAN ID. The allocation of CAN IDs is not standardized, meaning that manufacturers can create their own mapping of CAN IDs and the byte locations in a CAN message to outline what information is stored in what type of message within their CAN network. In practice, this results in every vehicle manufacturer potentially using a different CAN ID and byte index to store the engine coolant temperature, for example. Note that manufacturers may also duplicate certain values, i.e., the engine coolant temperature may be stored in CAN frame with ID 0x288 byte 1, as well as in CAN ID 0x420 byte 4 [7].

Hence, the lack of generalizablility and the fact that the needed mappings are not public information are the reasons that, at this time, the contents of a CAN frame are not an attractive feature for IDSs. In automotive networks, CAN frames with a specific CAN ID are often

recurrent, and their frequency is static. Current automotive IDSs often use characteristics based on this principle, such as inter-frame timing, as a feature for intrusion detection. This is based on the notion that when frames are modified or inserted, the average interval between two frames of the same CAN ID changes, which can be detected by an IDS [2, 3, 4, 5].

B. On-Board Diagnostics II (OBD-II)

The On-Board Diagnostics II (OBD-II) interface is a protocol that provides access to the status and error codes of various vehicle sub-systems. It is mostly used by automotive mechanics in vehicle diagnostics and emissions tests and can be accessed through the OBD-II port that is located in most vehicles near the driver's seat [8]. The protocol has been mandatory in the European Union for all gasoline cars since 2001, and since 2003 for all diesel cars [9]. Next to emissions-related information and stored vehicle trouble codes, the OBD-II interface provides diagnostics data that is obtained from the engine control unit, which is an ECU that is specifically concerned with engine operation. The Society of Automotive Engineers (SAE) Standard J1979 defines a method of requesting so-called parameters from the engine control unit through the OBD-II interface using *Parameter IDentification numbers* (PIDs) [8]. This allows for the interrogation OBD-II system for a specific OBD parameter from the engine control unit, such as the engine coolant temperature. The physical OBD-II port is comprised of 16 pins, of which 7 pins are not standardized and are therefore left to the manufacturer's discretion. As OBD-II is in essence the higher level protocol, it relies on lower level signaling protocols, of which five are standardized. One of these signaling protocols is ISO 15765-2 CAN, which is signaled through pins 6 (CAN-High) and 14 (CAN-Low) of the OBD-II port (see Figure 1) [8, 10]. On many vehicles, the CAN bus is therefore directly accessible through the OBD-II port of the vehicle. However, newer vehicles often utilize multiple, separate CAN buses that are used for different applications. For example, a vehicle may be equipped with a high-speed CAN bus for the powertrain and engine, and another, slower speed CAN bus that relays information from and to driver comfort systems such as infotainment and windscreen wipers. Newer vehicles therefore often contain a *CAN gateway* that interconnects these CAN buses and provides the connection to the OBD-II port as the OBD-II port is also used for manufacturer-specific diagnostics [11]. This results in the internal vehicle CAN bus not being directly accessible through the OBD-II port. Considering that our research requires direct access to the vehicle's powertrain CAN bus, our research has been limited to older vehicle models. Note that it is still possible to access a vehicle's powertrain CAN bus by 'tapping into' the twisted pair wires that are used for CAN communication. Because we did not want to damage our test vehicles, we have not attempted this in our research.

III. Related work

Our research aims to extend the work of Kang et al. [6]. Their process of automated reverse-engineering uses two methods to match the value that is returned from an OBD-II interrogation to a value located in CAN data that is transmitted around the time of the OBD-II interrogation. The first method uses a direct one-on-one search, meaning that the algorithm searches for the exact byte values that are returned by the OBD-II interrogation, in little endian, as well as in big endian byte order. If a match is found, the CAN frame that contains the value is added to a list of candidate CAN frames. If, at the end of this process, the list of candidate CAN frames contains multiple candidates, the second method is applied to eliminate candidates until a final candidate is left. This second method consists of repeatedly interrogating the OBD-II system. The return value is then used to check whether the during that interrogation recorded CAN frames of the same CAN ID as the initial candidate still contain the value that was returned by the OBD-II system in this interrogation at the same byte location. This process is repeated until one candidate remains. The byte location of the remaining candidate that has repeatedly matched the OBD-II parameter is then regarded as related to that OBD-II parameter. Their method therefore relies on the following assumptions:

- If multiple candidates are found with the first method, then some of these candidates were found by coincidence; i.e., the value returned by the OBD-II interrogation matched a value in a CAN frame at random.
- The probability that subsequent OBD-II interrogation values match the same CAN frame at random is extremely low; thus, if an OBD-II interrogation repeatedly matches a value in a CAN message, it can be concluded that these values are related.

For each match between OBD-II parameter and CAN frame, the researchers performed a validation by constructing a new CAN frame with an artificial value at the byte index that matched the OBD-II parameter, and sent this CAN frame onto the vehicle's bus. The researchers then observed the vehicle's response. For example, for the OBD-II 'engine RPM' parameter, the researchers observed that when a new CAN frame was sent with an artificial value at the byte index that matched the engine RPM, the vehicle's RPM gauge in the dashboard moved up and down.

IV. Research Question

The main question for this research is defined as:

To what extent can we reverse-engineer CAN messages using OBD-II interrogations and correlation coefficients when a translation is used?

V. Methodology

In this section we will detail the approach taken in performing this research. We will discuss the theory behind our approach, the practical implementation, detail the design of our proof-of-concept code, explain our method for reverse-engineering the CAN formulas and discuss design choices and limitations of our method.

A. Theory

As briefly discussed in Section I, our research aims to extend the approach used by Kang et al. in order to be able to discover in what CAN frames values are stored that we can request through the OBD-II interface when a translation on the CAN value is used by the vehicle manufacturer. Comparable to their approach, we interrogate the OBD-II interface for a specific parameter, and at the same time, we record the CAN frames that are sent over the bus. However, instead of matching the value obtained from the OBD-II interface to values in the recorded CAN frames directly, we perform the OBD-II interrogation and CAN recordings repeatedly. From these OBD-II interrogations and corresponding CAN bus recordings, we obtain two data structures. The first data structure is a list that contains the OBD-II responded values, and thus the length of that list corresponds to the number of performed interrogations. The second data structure is a list of two-dimensional arrays. Each entry in this list corresponds to a recording of the CAN frames that were sent over the bus during one interrogation. The two-dimensional array contains rows of CAN frames that consist of the CAN ID for that frame and the actual data, an array of at most 8 bytes. Considering that our objective is to discover whether a value in a CAN frame is related to the value returned by the OBD-II interrogation, we check whether a correlation exists over a number of interrogations between the OBD-II values and the values that occur in a CAN frame with a specific CAN ID at a specific byte location. As outlined in Section II-A, CAN frames with a specific CAN ID are often recurrent, and therefore, multiple CAN frames with the same unique CAN ID can occur in one recording. As we use the Pearson Correlation Coefficient (PCC) to determine whether a relation exists between

the OBD-II values and the data in a CAN frame, the input data series must have the same dimensions (i.e., the sample size n must be equal). Therefore, for each CAN frame with ID a, and therein for each byte index b, we compute the average value of that data within one recording. For all recordings (interrogations) together, this yields a vector of which the length is equal to the number of interrogations, for each pair (a, b), which we use to compute the PCC between the OBD-II values and the CAN data. This is illustrated in Figure 3 in the Appendix.

The PCC is computed using the paired data $\{(x_1, y_1), ..., (x_n, y_n)\}$ where $x_i \in X$, $y_i \in Y$ and where $X =$ the list of values retrieved through the OBD-II interrogations, and $Y =$ the list of averages per interrogation for a specific CAN ID a at byte index b within that CAN ID. The PCC can then be computed using the following formula:

$$r_{xy} = \frac{\sum_{i=1}^{n}(x_i - \bar{x})(y_i - \bar{y})}{\sqrt{\sum_{i=1}^{n}(x_i - \bar{x})^2}\sqrt{\sum_{i=1}^{n}(y_i - \bar{y})^2}} \quad (1)$$

where n is the sample size, and $\bar{x}$ and $\bar{y}$ are the sample means for X and Y, respectively.

B. Experiment setup

Our experiment required us to capture data on the CAN bus and do OBD requests at the same time. To accomplish this, we connected a PiCAN 2 CAN bus board to a Raspberry Pi 4 model B. This board in turn allowed us to connect a DB9 to OBD-II cable from our Pi to the test vehicles. The PiCAN only supports reading CAN and not OBD messages and each vehicle only has one OBD port, so to be able to do OBD-II interrogations simultaneously we had to solder the CAN-high, CAN-low and ground pins of the DB9 to OBD-II cable to an ELM327 generic OBD-II microcontroller, as can be seen in figure 1. We ran all our tests on two test vehicles, an Audi A4 B7 from 2006 and a Hyundai i10 from 2007.

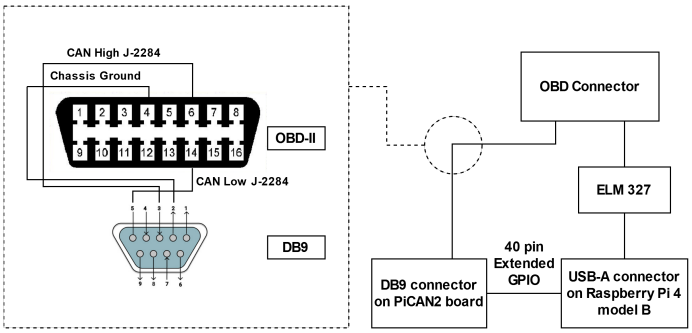

Fig. 1. This schematic shows how both the PiCAN2 board and the ELM327 were connected to the same OBD port. The pinout of the ELM327 is not shown, as it is connected on all pins and selects which ones to use based on the specific version of the OBD protocol used by the test vehicle.

C. Proof-of-concept

The proof-of-concept was programmed in Python 3, as it is the latest version and supports CAN and OBD libraries.[1] We also incorporated multi-threading, to be able to obtain the CAN data asynchronously during the OBD interrogations. The code was separated into four main phases and operated as follows:

1) Retrieve a list of supported PIDs. This is a function of the OBD library and avoids having to waste time and computing power on values the test vehicle in question can not return.
2) Capture the CAN data and do an OBD interrogation simultaneously for the selected PID.
3) Compute the averages of each unique CAN ID, for each byte index for each interrogation.
4) Calculate the correlation over the resulting OBD and CAN data series where the length of these one dimensional arrays is equal to the amount of interrogations. The result, consisting of each CAN ID and byte index pair, their correlation to the corresponding OBD value and the OBD and averaged CAN data, is then saved to a CSV file.

We perform steps 2-4 for each OBD-II PID.

D. Least-squares fitting

After we had found the CAN ID and respective byte index that correlated with the requested OBD value, we used the Scipy Python library to fit a least-squares line function $y = ax+b$ to the data for the CAN ID-byte index candidates that showed a high correlation with the OBD data, in order to be able to deduce a translation formula from CAN to OBD. We assumed a linear $y = ax + b$ function as we have observed this to be the only formula used from preliminary tests, which were in line with the formulas in [7]. We did not consider other possible formulas, because we assume that the goal of using a translation is not obfuscation, but that the goal is rather to be able to fit certain human-readable values in the constraints of an 8-bit integer.

E. Design decisions and limitations

Considering that our experiments contain many variables, we had to make some design choices that could influence the outcome. We further detail these decisions below and mention their limitations.

Due to the scope of this research, we opted for a simple and practical test setup. We therefore tested our solution by connecting our DB9 to OBD-II cable to the OBD port of the test vehicle and launching our script. Since some and especially modern vehicles have a (security) gateway behind the OBD port, as explained in Section II, this approach will not always work. Although the CAN buses of these vehicles rely on the same basic principles, we would have to cut CAN wires in the dashboard to have access to the CAN bus.

We experimented with different testing procedures, doing our tests when the test vehicle was just started up and after 15 minutes of warming up, as well as stationary and on the road. We settled on running our tests whilst driving after letting the car idle for 15 minutes. We did two experiments for each vehicle, testing with both 100 and 200 interrogations. These numbers were mostly chosen for practical reasons, as it allowed us to get data sets that are as large as possible whilst staying within a realistic time frame. Choosing two different amounts of interrogations also allowed us to see the influence of an increased amount of interrogations on the accuracy of our method. A more in-depth analysis on the relevancy of the procedure can be found in Sections VII and IX.

We have opted to display all CAN ID-byte index combinations that had a correlation with the OBD data series of more than 0.9. We believe setting a threshold is a more truthful representation than only storing the CAN ID and byte index with the highest correlation, since multiple CAN IDs and indices can contain the same information, as stated in Section II. This specific value was selected by analyzing the results of our preliminary testing and since in the real world it is possible to have high correlation on random data and to avoid false positives, this value had to be high. We do not expect the correlation to be 1, since the resolution of our CAN data is lowered by taking averages and therefore a perfect match should only occur if the data has almost no fluctuation.

[1]Our proof-of-concept code can be found on the OS3 Gitlab server. It is available at the following url: https://gitlab.os3.nl/bblaauwendraad/rp2—rp103.git

It is important to note that due to the proprietary nature of the CAN bus, we can not truly claim that our findings are correct unless we get confirmation from the manufacturer. However:

1) We found a source that corroborates our findings, as mentioned in Section I.
2) The correlation proves that the data series correspond to each other.
3) We have sent CAN messages back to the car and we have drawn conclusions from the resulting events. The CAN messages we sent back were generated using values comparable to the original frames to avoid damaging the test vehicle.

VI. Results

In this section, we will elaborate on our results. We have identified three subsets of results, PIDs with a high correlation match, PIDs with an ambiguous result and PIDs with no match at all. Lastly, we will also highlight some notable findings that we observed.

The data that we obtained differed between the two vehicles in the number of CAN frames that were sent per interrogation. For the Audi vehicle, we recorded approximately 600 frames per interrogation, whereas the Hyundai vehicle only transmitted around 30 frames per interrogation. Due to the implementation of our program, a lower number of CAN frames per interrogation results in averages that are possibly more true to the real value at the exact time of the interrogation.

We have only selected the OBD-II PIDs that returned a continuous value, i.e., PIDs for which a request would not return a discrete state such as 'on' or 'off'. The Audi vehicle supported 8 OBD-II PIDs, while the Hyundai vehicle supported 18 OBD-II PIDs. We have therefore obtained more results from the Hyundai vehicle.

A. High correlation

Our results show that in both vehicles, there is data from multiple PIDs that correlates highly with data averaged from specific CAN ID-byte index pairs in both 100 and 200 interrogations. With these PIDs, the highest correlating CAN ID and byte index of both 100 and 200 interrogation runs are identical. In other words, the larger data set did not yield different results for PIDs that have a high correlation with a specific CAN ID-byte index pair. A selection of these results is shown in Table I.

B. No match

Our results also show that there are OBD PIDs for which no match can be found, i.e., the correlation between the OBD values and the values from a CAN ID-byte index pair is < 0.9. These results are outlined in Table II.

C. Ambiguous cases

As outlined in Section V, the experiments were performed once with 100 and once with 200 interrogations. A higher number of interrogations results in a larger data series that is used for correlation calculation. Concerning the number of interrogations, we have made a number of observations; these are also outlined in Table III. Firstly, there are PIDs for which a higher number of interrogations 'filters out' potential false positive CAN ID-byte index candidates, i.e., a higher number of interrogations yields a smaller number of candidate CAN ID-byte index pairs, that, according to [7], yields less false positive candidates than a lower number of interrogations. However, our results show that there are also PIDs for which a higher number of interrogations yields results containing a larger number of candidate CAN ID-byte index pairs. Lastly, our results show that for the Hyundai catalyst temperature PID, the run with 100 interrogations yielded no candidates, while the run with 200 interrogations produced a candidate with fairly high correlation, i.e., 0.95.

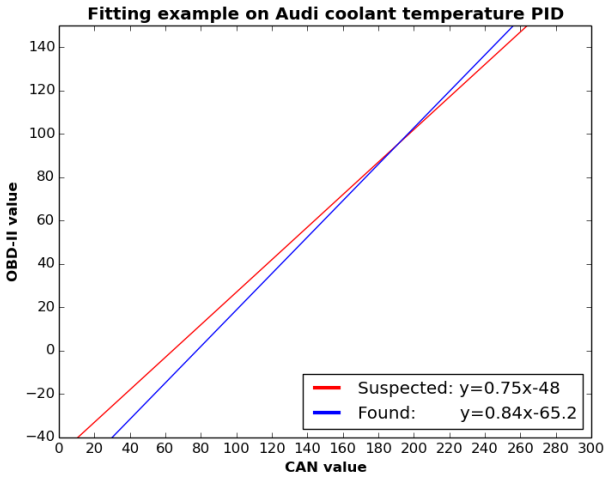

Fig. 2. The computed linear least-squares fitting function is shown in blue, while the suspected formula from [7] is displayed in red.

D. Least-squares fitting

Using Scipy's `curve_fit` function, we computed the optimal least-squares line function for the coolant temperature PID in the Audi vehicle. The coolant temperature was chosen as an example, however this method works for all PIDs with high correlation coefficients. The `curve_fit` function can be used to fit a least squared function to data, resulting in a curve (or line) formula that best fits that data. From [7], we know that the translation formula used is $0.75x - 48$ where x represents the decimal value in the CAN frame. The optimal fit function that Scipy returned approximated to $y = 0.84x - 65.2$. An illustration is provided in Figure 2.

E. Re-sending CAN messages

To reinforce our results, we have attempted to re-send CAN messages over the Audi vehicle's CAN bus to observe how the vehicle would react. We performed this for all PIDs that scored a high correlation on the Audi (see Table I). In the majority of cases, we observed no response from the vehicle, but for the RPM and engine coolant temperature PIDs, we observed that respectively the RPM gauge and the engine coolant temperature responded by going up to approximately the value that we put in the CAN frame that we sent. The re-sending of CAN messages to the test vehicle was only attempted for the Audi vehicle because this vehicle was owned by us.

VII. Discussion

In our testing, we noticed fluctuations in the performance of our approach. Increasing the amount of interrogations showed both positive effects on one PID and negative effects on the other. We also noticed different PIDs matching to the same CAN ID-byte index. We believe both anomalies to be due to two characteristics of our approach, which we will discuss below.

A. Characteristic

The first characteristic is how we compute the correlation. Seeing that multiple frames with the same CAN ID can be transmitted during one recording, we had to compute the average of these CAN ID-byte index pairs for each unique CAN ID and each interrogation. This way, we acquired two data series of identical dimensions. The more messages are sent on the CAN bus during the time of recording, the more resolution is lost through taking these averages.

Vehicle	PID	CAN ID (Hex)	Byte Index	Avg Correlation
Audi	Engine RPM	0x280	3	0.997
Audi	Intake manifold absolute pressure	0x588	4	0.999
Audi	Mass Air Flow (MAF) rate	0x288	6	0.960
Hyundai	Engine coolant temperature	0x329	1	0.971
Hyundai	Engine RPM	0x316	3	0.993
Hyundai	Vehicle speed	0x580	7	0.998
Hyundai	Calculated engine load	0x316	4	0.970
Hyundai	Absolute load value	0x316	4	0.991
Hyundai	Intake manifold absolute pressure	0x316	4	0.967
Hyundai	Relative throttle position	0x329	5	0.991

TABLE I
A TABLE SHOWING THE PIDS WITH HIGH CORRELATION. THE CID IS THE UNIQUE CAN ID WITH THE HIGHEST CORRELATION IN BOTH 100 AND 200 INTERROGATION RUNS IN HEXADECIMAL NOTATION. THE AVERAGE CORRELATION VALUE ROUNDED TO THREE DECIMALS IS GIVEN OVER BOTH RUNS.

Vehicle	**OBD-II PID**
Audi	Calculated engine load
Audi	Intake air temperature
Audi	Throttle position
Hyundai	Ambient air temperature
Hyundai	Fuel–Air commanded equivalence ratio
Hyundai	Commanded evaporative purge
Hyundai	Intake air temperature
Hyundai	Long term fuel trim - bank 1
Hyundai	Oxygen sensor 1
Hyundai	Oxygen sensor 2
Hyundai	Timing advance

TABLE II
A TABLE SHOWING THE PIDS FOR WHICH NO MATCH WAS FOUND.

As a result of this occurrence, CAN ID-byte index pairs that have a similar progression through the testing can score a very comparable correlation. A good example of this is the speed PID on the Audi, as it provides the correct CAN ID-byte index pair with a 100 interrogations, but finds a higher correlation for a different CAN ID-byte index pair at 200 interrogations.

The second characteristic is our testing environment and methodology. During normal driving conditions, it is not an uncommon occurrence that different PIDs have similar progression throughout the run. E.g. both the coolant temperature and oil temperature rise steadily after initial startup of the vehicle. This can be seen clearly on the Audi, as it matched on (what we presume to be) both the oil and coolant temperature CAN ID-byte indices. When we reran our experiments on that test vehicle after it had warmed up however, the oil temperature CAN ID-byte index pair no longer displayed a correlation of over 0.9. We suspect many PIDs to have a similar accidental correlation due to real world variables, e.g. the intake pressure and engine load and the speed and wheel rotation. We therefore argue that an extensive testing procedure, taking into account the unique characteristics of each PID interrogated, could greatly improve accuracy.

B. No matches

With regards to the PIDs which did not find a match at all, we have three possible explanations:

- The return value for the OBD interrogation is comprised of multiple CAN ID-byte indices. If these values are combined through a formula, our method will by definition not be able to find their correlation coefficient.
- For some PIDs, the return value is stored in multiple bytes, as it is too large fit to in a single byte. The paper by Kang et al. shows that this is the case with the fuel pressure, for example. Detecting values across multiple byte indices was out of scope for our research.
- Some PID values (almost) never change. This can be seen on e.g. the timing advance PID of the Hyundai. Since there are multiple CAN ID-byte index pairs on the CAN bus that contain only zero's, a perfect correlation coefficient between these two data series will be found. Furthermore, when two data series are exactly the same, the correlation coefficient will be n/a in our program and not show up in the results.

The limitation that one on one values can not be found is deliberate, as our research is an addition to the work by Kang et al. and not a replacement. Our method is designed to find CAN ID-byte index pairs that used translation on the original values and will never perfectly match due to our resolution decrease on the CAN data by calculating the averages.

C. Least-squares fitting

As can be seen by the fitting example in Section VI, we were not able to find the exact formulas used on the CAN values. We think this is both due to the correlation being imperfect and our resolution loss through taking averages of the CAN data. However, the formula found is a close approximation, especially when the expected range of the PID in question is known. We argue that this information is therefore still useful for IDS development, as the goal of knowing the formulas is to be able to determine if a value is within a expected range and changes at a expected rate. E.g. when the coolant temperature increases from 30 (value after conversion with the found formula) to 250 in a relatively short amount of time, it would indicate that either a sensor is defective or an attacker is inserting CAN frames on the bus. An IDS could detect such a implausible fluctuation and trigger an alarm.

VIII. CONCLUSION

Our research aimed to extend the approach by Kang et al. to be able to match CAN ID-byte index pairs to OBD-II PIDs when a translation is used in the CAN data. We argued that the correlation coefficient between the CAN data and the OBD data can relate both data series to each other in order to discover which CAN ID and byte index stores data that is directly related to the OBD-II PID. Our results have shown that, when comparing against [7], we can correctly match CAN ID-byte index pairs to OBD-II PIDs on multiple vehicles with some practical limitations. This research may prove useful for future IDS development that intends to implement plausibility checks on data as an IDS feature.

IX. FUTURE WORK

Due to both time and budgetary constraints, and practical limitations due to the ongoing COVID-19 pandemic, we were not able to do our research as extensively as we hoped. Furthermore, we came to many interesting insights during our study of related work and our own experiments, which warrant further examination. These suggestions for future work can be found below.

Vehicle	PID	Ambiguity | int. = interrogations
Audi	Engine coolant temperature	100 int.: returned some false positives (e.g., engine oil temperature) 200 int.: yielded two correct CAN ID-byte index pairs according to [7]i.
Audi	Vehicle speed	100 int.: returned 8 candidates 200 int.: returned 10 candidates
Hyundai	Throttle position	100 int.: returned three candidates 200 int.: resulted in one candidate
Hyundai	Intake manifold absolute pressure	100 int.: returned three candidates 200 int.: yielded two candidates

TABLE III
AMBIGUOUS PIDS

A. Sample size

In this paper, we emphasize the usefulness of the correlation method proposed because it is generalizable. We were however only able to confirm our theories on two test vehicles, seeing as we opted to only connect to the CAN bus over the OBD-II port as opposed to splicing into the CAN-High and CAN-Low wires of the CAN-busses directly to avoid damaging the test vehicles. This meant we were limited to vehicles which have their power-train CAN bus directly connected to the OBD port. Especially on more modern vehicles, this is no longer the case, as they use a (security) gateway. Testing our theory on a large amount of test vehicles from different brands, construction years and models would greatly improve both the relevance and usefulness for IDS development of our proposed method.

B. Proof

We were not able to conclusively prove the correctness of our assumptions. As detailed in Section V, we assume all CAN ID and byte index pairs with a correlation of over 0.9 correspond to the OBD value requested. We tried to confirm our assumptions by sending CAN frames with the same CAN IDs and altered byte values on the CAN bus and monitoring the resulting events and comparing our findings with findings of car enthusiasts online. This is not definitive proof. We speculate that taking a look at the circuitry of the sensors and actuators that are sending these messages on the CAN bus could confirm our findings. The most reliable solution to the problem would be getting confirmation from the manufacturer of the vehicle in question. Obtaining this information could also be used to check the correctness of the formulas we reverse-engineered using fitting. We highly doubt however that these companies are willing to disclose proprietary information and found this to be out of our scope.

C. Testing procedure

During our testing, we found that a custom testing procedure for each PID could greatly improve the accuracy of our method. Designing such a testing procedure would require adjusting (at least) the testing environment, amount of interrogations and vehicle operation. For example, in our testing, we found that it yields a more accurate result to test the coolant temperature when the car is on the road as opposed to stationary, and warmed up as opposed to just started. The waiting ensures that the coolant temperature is not warming up in a linear fashion identical to the oil temperature and the driving ensures the value still fluctuates enough to obtain an accurate correlation. The impact of code execution times on the result can also be tested, e.g. by adding an artificial delay to see the influence on the results of capturing more CAN frames per interrogation.

D. Performance

As this paper contains academic research and our code is a proof-of-concept, performance was not considered. However, one of the main benefits and reasons for our approach is that it helps IDS development. We also assume our method to be generalizable which saves IDS developers to manually obtain the CAN ID-byte index to OBD value mapping when working with vehicles of different manufacturers, makes and models. Therefore, a focus on improving speed and performance of our code could prove useful. A faster program will also result in a shorter CAN recording window, which lessens the resolution decrease that occurs when computing averages over the CAN data, potentially further increasing the accuracy. The same effect could be achieved through running on more powerful hardware.

X. ACKNOWLEDGEMENTS

We would like to thank our supervisors, Ruben Koeze and Sander Ubink, for their assistance and guidance during this project. We are particularly grateful to Marco Rajk for his expertise on automotive CAN bus systems and the time he has taken to help us. We would like to express our gratitude to Jan van Aller and Marthe de Vries for providing one of the test vehicles.

REFERENCES

[1] *CAN Specification, version 2.0*. Tech. rep. Stuttgart, Germany: Robert Bosch GmbH, 1991. URL: http://esd.cs.ucr.edu/webres/can20.pdf.

[2] Michael R. Moore et al. "Modeling Inter-Signal Arrival Times for Accurate Detection of CAN Bus Signal Injection Attacks: A Data-Driven Approach to in-Vehicle Intrusion Detection". In: *Proceedings of the 12th Annual Conference on Cyber and Information Security Research*. CISRC '17. Oak Ridge, Tennessee, USA: Association for Computing Machinery, 2017. ISBN: 9781450348553. DOI: 10.1145/3064814.3064816. URL: https://doi.org/10.1145/3064814.3064816.

[3] M. Gmiden, M. H. Gmiden, and H. Trabelsi. "An intrusion detection method for securing in-vehicle CAN bus". In: *2016 17th International Conference on Sciences and Techniques of Automatic Control and Computer Engineering (STA)*. 2016, pp. 176–180.

[4] Kyong-Tak Cho and Kang G. Shin. "Fingerprinting Electronic Control Units for Vehicle Intrusion Detection". In: *25th USENIX Security Symposium (USENIX Security 16)*. Austin, TX: USENIX Association, Aug. 2016, pp. 911–927. ISBN: 978-1-931971-32-4. URL: https://www.usenix.org/conference/usenixsecurity16/technical-sessions/presentation/cho.

[5] Noräs Salman and Marco Bresch. "Design and implementation of an Intrusion Detection System (IDS) for in-vehicle networks". MA thesis. Chalmers University of Technology / University of Gothenborg, 2017. URL: https://odr.chalmers.se/bitstream/20.500.12380/251871/1/251871.pdf.

[6] T. U. Kang et al. “Automated Reverse Engineering and Attack for CAN Using OBD-II”. In: *2018 IEEE 88th Vehicular Technology Conference (VTC-Fall)*. 2018, pp. 1–7.

[7] OpenStreetMap. *Audi A4 B7 Known CAN IDs*. URL: https://wiki.openstreetmap.org/wiki/VW-CAN.

[8] *SAE J1979 E/E Diagnostic Test Modes / ISO 15031-5:2015 Road vehicles - Communication between vehicle and external equipment for emissions-related diagnostics - Part 5: Emissions-related diagnostic services*. Standard. Troy, MI, United States of America / Geneva, Switzerland: Society of Automotive Engineers (SAE) / International Organization for Standardization (ISO), Feb. 2012.

[9] Council of European Union. *Directive 98/69/EC of the European Parliament and of the Council*. 1998.

[10] *ISO 15765-2:2016 - Road vehicles — Diagnostic communication over Controller Area Network (DoCAN) — Part 2: Transport protocol and network layer services*. Standard. Geneva, Switzerland: International Organization for Standardization (ISO), Apr. 2016.

[11] *Fahrzeugdiagnose-, Mess- und Informationssystem VAS 5051 - Aufbau und Funktionen - Selbststudienprogramm Nr. 202 / Vehicle Diagnosis, Measuring and Information System VAS 5051 - Structure and Functions - Self Study Program No. 202*. Technical Report. Volkswagen AG.

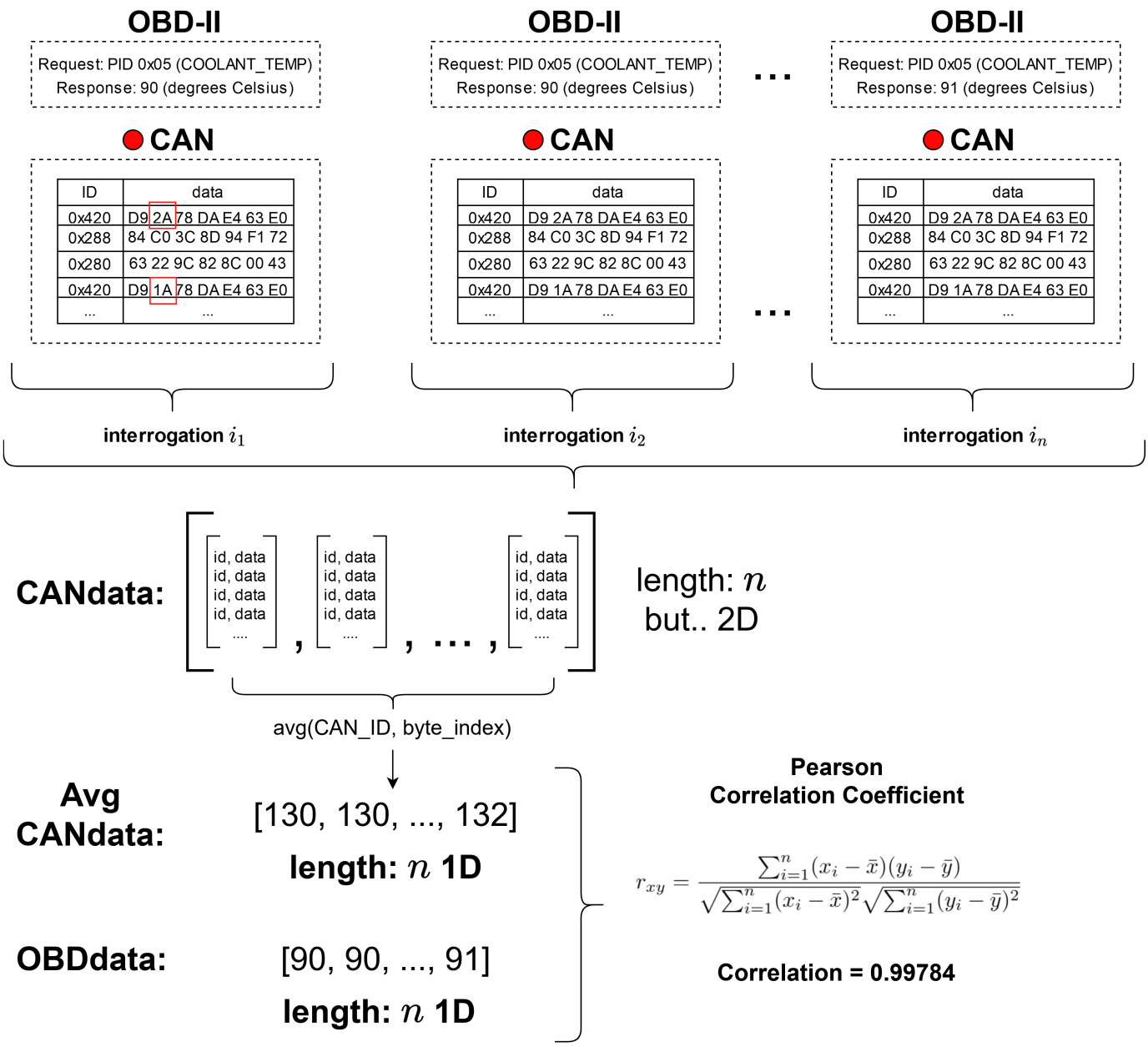

Fig. 3. An illustration of the approach that we used to calculate the averages and obtain the Pearson correlation coefficient for each CAN ID-byte index pair.