An Enhanced Method for Reverse Engineering CAN Data Payload

Wonsuk Choi, Seyoung Lee, Kyungho Joo, Hyo Jin Jo, and Dong Hoon Lee, *Member, IEEE*

***Abstract*—Recently, numerous electronic components are installed in vehicles, providing drivers and passengers with increased safety and convenience. The electronic components construct an in-vehicle network that internally shares relevant status information about the vehicle. As modern vehicles become more computerized, the potential for automotive cyber-security threats also increases a fact that has been illustrated clearly by various car-hacking demonstrations. Using the controller area network (CAN), the de facto standard protocol in the automotive industry that facilitates in-vehicle network communication, car-hacking demonstrations inject critical CAN messages to control vehicular functions. In efforts to address this security issue, car manufacturers, in turn, have made confidential the CAN database (i.e., DBC format file), where signal information assigned in the CAN data payload is specified. However, it has since become known that this policy does not hermetically seal a vehicular network against cyber attacks. On the contrary, in-depth automotive security research has been hindered significantly because of the limited information accessible by researchers. For example, automotive intrusion detection systems (IDS) identify and alert when there is a vehicular break-in, and this technology is a major area of study in automotive cyber security research. For the automotive IDS that analyzes CAN traffic, information in the DBC format file greatly improves detection veracity. However, most IDS technologies to date have been independently developed without the confidential CAN DB information and, as a result, do not mitigate threats to a satisfactory standard. In this paper, we propose an enhanced method that identifies signal boundaries in a CAN data payload, which is specified in the DBC format file. Unlike an existing method that is designed based on total bit-flip rates, our method analyzes bit-flip time series not total bit-flip rates so that signal boundaries can be more clearly identified. In this paper, we use a publicly available DBC format file called OpenDBC as a reference, and show that our method outperforms the existing method.**

***Index Terms*—Automotive security, CAN DBC format file, controller area network (CAN), reverse engineering.**

Manuscript received March 27, 2020; revised November 14, 2020; accepted February 18, 2021. Date of publication March 3, 2021; date of current version May 5, 2021. This work was supported in part by the Samsung Research Funding, and Incubation Center for Future Technology under Project No. SRFC-TB1403-51, and in part by the National Research Foundation of Korea (NRF) grant funded by the Korea government (MSIT) (NRF-2020R1C1C1007446). The review of this article was coordinated by Prof. F. Lavagetto. *(Corresponding author: Dong Hoon Lee.)*

Wonsuk Choi is with the Division of IT Convergence Engineering, Hansung University, Seoul 02876, South Korea (e-mail: wonsuk85.choi@gmail.com).

Seyoung Lee, Kyungho Joo, and Dong Hoon Lee are with the Graduate School of Information Security, Korea University, Seoul 02841, South Korea (e-mail: seyoung0131@korea.ac.kr; wnrudgh16@korea.ac.kr; donghlee@korea.ac.kr).

Hyo Jin Jo is with the School of Software, Soongsil University, Seoul 06978, South Korea (e-mail: hyojinjo86@gmail.com).

Digital Object Identifier 10.1109/TVT.2021.3063261

I. Introduction

In THE automotive industry, autonomous driving capabilities and connected vehicles are providing drivers and passengers with enhanced driving experiences. To achieve these impressive functions, a number of electronic control units (ECUs) are installed in vehicles to form an in-vehicle network. While computerized systems enhance safety and convenience, automotive cyber security threats are on the rise. In recent years, a number of studies have highlighted the findings of car-hacking demonstrations to this effect [1]–[5]. The controller area network (CAN) protocol, the de facto standard for in-vehicle network communication, is where the bulk of ECUs are connected and share relevant status information about the vehicle. The demonstrations were conducted by injecting meaningful CAN messages into the in-vehicle CAN network, which resulted in electronic hijacking of certain vehicular functions. It is highly expected that car-hacking attempts will increase in the real world as vehicles are increasingly connected over complex external networks, such as the Internet or cellular networks.

To prevent attackers from finding CAN messages that control critical commands (CAN messages), car manufacturers have made confidential the information in CAN database (i.e., DBC format file). Superficially, this policy seems to provide air-tight protection, but it has been proven possible to infer partial CAN database (DB) information since the CAN traffic is not encrypted. Magnifying the danger of this flaw, today it is easy for anyone to learn how to analyze CAN traffic on the Internet with ubiquitous tools like YouTube [6]. Contrary to its purpose to prevent cyber attacks, this policy has, rather, created large hurdles for security researchers, who have only been able to develop subsequent security protocols based on imperfect information. One area directly affected by this is automotive intrusion detection system (IDS) technology, which is a major interest field in automotive security studies. [7]–[14]. An automotive IDS detects an in-vehicle intrusion by analyzing CAN traffic, but, unfortunately, all of these technologies to date have been developed with information only publicly available. As a result, these systems marginally improve security, as they cannot detect complicated attacks on in-vehicle CAN networks perfectly. Beside automotive IDS, fuzzing, which provides invalid or random data as input to a target system, has recently been applied to the CAN bus [15], [16]. Fuzzing testing on CAN traffic functions as an automated software-testing technique that finds unexpected vulnerabilities in ECU software. However, there is no way to efficiently perform fuzzing on ECUs because of the undisclosed information hidden in the DBC format file. Furthermore, a brute-force or random search without the DBC

format file cannot efficiently locate vulnerabilities because the number of possible input cases is overwhelming.

To mitigate this, some studies have attempted to reconstruct a DBC format file [17]–[19]. These attempts can be divided into two categories of invasive and non-invasive methods. The invasive manner involves vehicles being disassembled to physically access an ECU and analyze its firmware (i.e., reverse engineering) [19]. Since the invasive manner can be simply blocked by any preventative method, such as disabling a debugging port, the non-invasive manner may be more desirable for reconstruction. Accordingly, in this study, we focus on a non-invasive manner that allows for the DBC format file to be analyzed using only CAN traffic traces [17], [18]. Prior to our research, Markovitz and Wool [18] first presented a method to discover semantically meaningful signals conveyed in the data payload of a CAN message. However, the evaluation in that study is insufficient because it is based on simulated CAN traffic traces rather than real-world traces. Following this, Marchetti and Stabili [17] showed that the [18] method does not work correctly on real CAN traffic, and presented a new method called READ based on total bit-flip rates at each bit location in a data payload. During our investigation, we also found that the READ method has a limitation when it comes to identifying signal boundaries despite the improvement of analyzing real-world CAN traffic traces. Accuracy of the READ [17] method depends on how CAN traffic is measured, which implies that it is not practicable in wider applications. In this paper, we summarize the key elements of READ [17] and present a counter example to demonstrate cases in which READ is not able to correctly identify signal boundaries. We then present an enhanced method to identify signal boundaries in a CAN data payload. Our core concept is to create bit-flip rate time series for a single bit location and analyze the correlation between two consecutive bit locations. We show that our method outperforms the state-of-art READ method in identifying signal boundaries with the aid of the public domain DBC format file OpenDBC as a reference for signal boundaries. A detailed discussion of our contributions is as follows.

Our Contributions

- With the dataset from the real vehicles, we show examples to demonstrate cases in which READ [17] is not able to correctly identify signal boundaries. Unlike READ, our method analyzes bit-flip rate time series not total bit-flip rates so that our method is able to correctly identify signal boundaries that READ would fail to identify.
- Our method enables a dataset to be easily measured from a vehicle. The method to analyze total bit-flip rates has different results, depending on ways to measure a dataset. For example, a function is activated during a short period of time. The total bit-flip rates of the corresponding signal would be almost zero; accordingly, the signal would not be identified by READ.

II. Related Works

In this section, we describe related works to facilitate better understanding of how automotive cyber security research has been conducted with limited information about CAN DB. Cyber attacks on modern vehicles are executed by injecting forged and malicious messages in the CAN bus network [1]–[3], [20]. To prevent these attacks, some studies have aimed to improve the security of communication over the CAN bus by applying cryptographic protocols [21], [22]. However, these methods require modification of current ECU software and negatively impact CAN communication by, for example, imposing a heavy CAN buslaod or creating a low response time. Accordingly, automotive IDS quickly became a promising method to bridge the gap, as these frameworks do not require any modification of existing vehicular components. The self-adapting nature of automotive IDS allows for easy application in the CAN bus of actual vehicles, and hence, methods for automotive IDS have been extensively studied [7]–[14]. Following this, the concept of in-vehicle intrusion detection was first introduced by [7], where the characteristics of intrusion detection patterns were presented. Because the formal specifications of CAN communication is confidential, all the characteristics mentioned in [7] are publicly accessible. As a result, most existing research cannot fully analyze CAN traffic because full access to the full DBC format file information is impossible. Next, we describe how researchers have developed methods with only this limited information.

The simplest approach is to check legitimate CAN IDs used in the CAN bus [8]. Because the number of legitimate CAN IDs is relatively small (usually less than 100), malicious CAN messages with invalid CAN IDs can be easily detected. However, this approach is also easily evaded by injecting messages with valid CAN IDs and forged data payloads. Transmission periodicity is also one of the characteristics used in automotive IDS technologies. Since most CAN messages are periodically transmitted on the CAN bus, the injection of malicious CAN messages clearly increases transmission frequency [9], [10]. Still, transmission-frequency-based methods are unable to detect injections by an advanced adversary who successfully mimics the original message frequency. Indeed, Greenberg [23] succeeded in disabling preventive measures based on message frequency and successfully transmitted malicious CAN messages, causing the vehicle to malfunction [11]. Moreover, the transmission periodicity of CAN messages is not constant in practice, as frequencies of periodic CAN transmissions extracted from real CAN traffic often exhibit unexpectedly high variability, which leads to unacceptably high error rates.

To address the limitations of message-frequency-based automotive IDS frameworks, behavioral characteristics were examined to detect vehicular intrusions [11]–[14]. To start, ECUs can be correctly identified based on their unique behavior characteristics. A pair with an invalid match between a CAN ID for a sender and an identified ECU are considered an anomaly. Murvay and Grozai [14] proposed a method to examine the electrical characteristics of CAN messages generated from ECUs (i.e., CAN controllers). However, this method cannot be applied to a real-world CAN bus because the authors failed to consider the situation in which multiple ECUs simultaneously transmit CAN messages. In the arbitration decision, multiple ECU simultaneously generate electric signals, and hence, unique characteristics become blended. Choi *et al.* [12] subsequently proposed a method to examine the electrical characteristics of CAN messages for ECU identification. Compared with [14], Choi *et al.* [12] designed a method that considers the arbitration decision process, producing a method that correctly identified

ECUs with high accuracy. However, this method required the use of an extended frame format to enable the additional 19-bit extended identifier field, whereas the standard format is commonly applied in actual vehicles. This implies that their method requires a modification of firmware in existing ECUs, which is infeasible in practice. Alternatively, Cho and Shin [11] suggested a method that does not require any modification of existing ECUs. To identify ECUs, the authors leveraged the fact that each ECU has a unique clock skew, even if these ECUs transmit messages within the same period. Unfortunately, this method can be also evaded by emulating the ECU clock skew as proposed in [24], in which clock emulation was demonstrated. This suggests that an attacker could inject malicious CAN messages without being detected. More recently, Choi *et al.* [13] and Kneib *et al.* [25] have both presented the enhanced automotive IDS that are designed based on electrical characteristics of CAN messages. However, these methods require special equipment to measure the electrical signals of CAN messages. Moreover, an embedded resource-constrained device must extract the features from a number of digitized samples, significantly increasing computational overhead.

Even though much research has been conducted to develop automotive IDS technologies, the main drawback shared by all the aforementioned works is that the methods were designed with very limited features that can be analyzed using public information, such as a generic CAN traffic trace. For example, the message arrival timestamp is the most common feature used in IDS frameworks, which is only able to detect simple attacks. Accordingly, most IDS frameworks having this limited information would most likely fail to identify advanced attacks, and hence, this is the main cause for high error rates among existing IDS frameworks. Clearly, this issue could be mitigated if access to the complete formal specifications of CAN messages was possible, including the list and all signal boundaries in the data payload, which is usually included in DBC format file. However, this information is presently accessible only by car manufacturers.

Beside automotive IDS technologies, the semantics of CAN messages would also be extremely applicable in automotive forensics using CAN traffic, which reconstructs the state of the vehicle before a collision. By analyzing CAN traffic measured at the time of an accident, the underlying cause could be easily identified from driver error to vehicle function failure or a cyber attack. To this end, Nilsson and Larson [26] first proposed a set of requirements for digital forensic investigations into cyber attacks on automotive in-vehicle networks. After that, Mansor *et al.* [27] proposed a reliable and secure privacy-preserving mechanism that efficiently built an automotive logging system specifically for forensics data collection and storage. These works in automotive forensics have helped other researchers understand the semantics of CAN messages. From a cursory glimpse at research, we can understand that some information about semantics and syntax of CAN messages is obtainable through invasive reverse engineering and manual analysis, as proposed in [2], [3] and [28].

In this paper, we present a unique algorithm to automate reverse-engineering of signals and their boundaries conveyed in CAN messages, which would contribute remarkably to automotive security research. We elaborate on our motivation for this study and the limitations of the existing method [17], [18] in the next section.

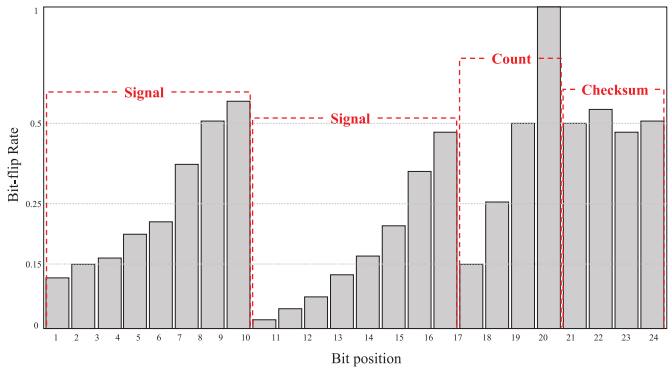


Fig. 1. The main concept of READ [17] is to identify signals and their boundaries based on bit-flip rates on each bit.

III. Problem Statement

To address security-related issues for the in-vehicle CAN bus, many researchers have attempted to analyze CAN traffic as a means to develop security methods [9], [11]–[13], [29]. However, these attempts failed to produce comprehensive analyses due to the confidential nature of DBC format files, a standard long maintained by car manufacturers. In academic research, this policy has created significant obstacles for further research on the CAN bus. Moreover, the intended effect of abating cyber attacks has yet to come to fruition. Conversely, a number of car-hacking demonstrations have proven successful despite not having knowledge of the formal specifications for CAN traffic. In effect, this policy has merely blocked the advancement of automotive security research.

On the other hand, the fuzzing technique that involves providing invalid or random data as input to a target system has been recently applied to the CAN bus [15], [16]. In theory, the fuzzing analysis on CAN traffic could be used as an automated software-testing technique to find unexpected vulnerabilities in ECU software. However, there is no way to efficiently perform fuzzing on ECUs because of the undisclosed information in the DBC format file. Although possible, a brute-force or random search without the DBC format file information is equally inefficient in finding vulnerabilities due to the large volume of input cases.

Despite this problematic confidentiality policy, security researchers have managed to conduct limited studies without precise information, with some studies even attempting to reconstruct DBC format files [17]–[19]. As mentioned before, READ [17] is a state-of-art framework that identifies signal boundaries assigned to the CAN data payload. However, we found that the accuracy of READ depends on how CAN traffic is measured, which implies that the method would not work in a general or wide application. We summarize READ and present a counter example in which READ is unable to correctly identify signal boundaries.

As shown in Fig. 1, READ computes total bit-flip rates on each bit position of data payloads for a particular CAN ID. Based on this total bit-flip rate, READ scans for pairs of consecutive bits

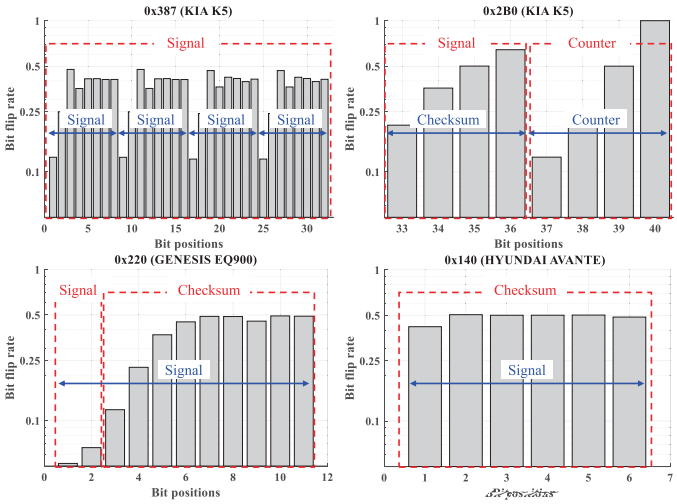


Fig. 2. Bit-flip rates from a real vehicle, which is a counter example of READ.

where the first bit has a rate higher than the second. Whenever a similar pair is found, a preliminary boundary is set between the two bits. The method is based on the assumption that drops in the bit-flip rate are caused by a less significant signal bit that is immediately followed by the most significant bit of an adjacent signal. In addition, the checksum signal also rests on the assumption that the bit-flip rate of all bits is distributed along a normal probability distribution centered on a rate of 0.5. However, we found that the shape of bit-flip rate signal pair is sometimes different from the expectation of READ. Fig. 2 shows the bit-flip rates for each bit that we measured from a real CAN bus of an actual vehicle (Kia Soul 2014). By the READ algorithm, signals and their boundaries should present like the figure in red, but the actual signals and their boundaries are shown in blue. Unlike the [17] expectation, the bit-flip rate for the checksum signal is not distributed according to a normal distribution. Moreover, a single signal for sensor values can be divided into two different signals.

IV. Backgrounds

In this section, we provide some background to inform a clearer understanding of our method.

A. In-Vehicle CAN Network

For driver safety and convenience, many functions of a vehicle are electronically controlled. A number of ECUs are installed in a modern vehicle to operate the digitized functions. ECUs periodically measure the status of a vehicle along with control some functions based on these measurements. For example, an engine ECU has a sensor to recognize when the accelerator is being pressed. Based on the measurement, the ECU actuates the engine by itself or delivers messages to another ECU to initiate the engine. This automated control process enables vehicles to operate more efficiently and safely, and more are being developed as time passes. In fact, today, luxury vehicles typically have 70 ECUs installed [5]. A set of ECUs construct an in-vehicle network where they communicate and, depending on similar functions, the in-vehicle network can be divided into sub-networks. For communication between ECUs

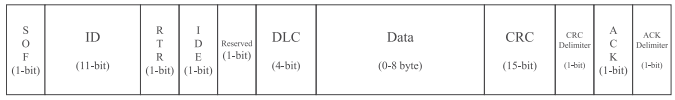


Fig. 3. Fields defined in a CAN data frame.

in the in-vehicle networks, a protocol is necessary, such as LIN, CAN, or FlexRay. Among these, the CAN protocol is the most widely used because it provides a bounded transmission delay among ECUs. In contrast, other protocols are usually used for infotainment or non-safety-critical system ECUs.

B. CAN Protocol

In the CAN standard, two different frame formats are defined: the standard frame format (described in CAN 2.0 A and CAN 2.0B) and the extended frame format (only described in CAN 2.0B). The standard frame format supports an 11-bit identifier, whereas the extended frame format supports a 29-bit identifier, which is made up of the 11-bit identifier and an 18-bit extended identifier. In general, the number of ECUs in a vehicle is less than 2^{11}, and they communicate using the standard frame format. In an in-vehicle CAN network, ECUs share data with each other using four different frame types: data, remote, error, and overload frames. We noted that CAN frame formats and CAN frame types should be distinguished. The common CAN messages that ECUs transmit are the data frame type. As shown in Fig. 3, a CAN data frame contains some fields such as identifier (ID), data length code (DLC), data, and cyclic redundancy check (CRC). The identifier field refers to the identifier of the transmitter rather than the receiver. This identifier represents its priority and a particular function, such as the engine temperature and the throttle valve angle. Generally, ECUs are assumed to not share the same identifier, and it is also assumed that one ECU has multiple identifiers. Commonly, vehicle manufactures put forth great effort to keep confidential which identifiers match which functions.

C. Structure of Data Payload Field

A CAN data payload consists of at least one signal. A signal is a piece of information transmitted by an ECU, such as vehicle speed. The length of and number of signals vary with CAN ID and are specified in the DBC format file for the corresponding vehicle. Without the DBC format file, our method is designed to identify the number of signals in a CAN data payload and their length by analyzing CAN traffics. Moreover, signals contain not only information but also other types of information [17], [18], as follows.

Physical value is a type of signal whose values indicate a particular physical status of a vehicle, such as wheel speed or steering wheel angle. ECUs are equipped with a number of sensors to measure the status of the vehicle and electrically control its functions based on the measurements. Accordingly, a number of signals for physical values are transmitted through an in-vehicle CAN network.

Counter is a signal whose value always increases by one with respect to the counter of the previous message. Counter enables liveness verification of the ECU transmitting CAN messages for a particular CAN ID. Moreover, a receiving ECU is able to

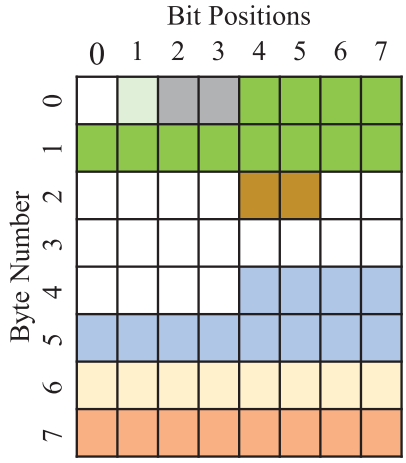


Fig. 4. Example of CAN signals.

recognize a retransmission of a CAN message that has already been received.

Checksum is a signal containing the checksum results for a data payload. Commonly, checksums detect transmission errors. Even though the cyclic redundancy check (CRC) field is standardized in the CAN data frame, some manufactures design an additional checksum field in the CAN data payload to provide stronger reliable CAN communication in their vehicles.

Unused / Constant is a signal that does not change. According to the DBC format file specification [30], the maximum and minimum values of a signal can be the same. This implies a constant. In addition, some bits in a data payload can be assigned as an unused field. Since our method is originally designed to identify signals based on bit-flip rates, it cannot distinguish between them. We consider these two different types the same.

An example is given in Fig. 4 showing multiple signals of different types, such as physical value, counter, checksum, unused bits, and constant bits. White regions are for unused bits which are not defined in the DBC format file.

V. Our Method

Our method identifies signal boundaries in a data payload that has at most a 64-bit length. Signal boundaries can be identified where trends of bit-flip rates significantly changes. Unlike the existing method, READ [17], which uses total bit-flip rates, our method analyzes time series of bit-flip rates by splitting the total CAN traffic into multiple fixed-size subsets. For a clear understanding of how to identify signal boundaries, we present our method in the following three steps.

A. Total Bit-Flip Rates

For given CAN traffic traces, our method extracts a subset, S_{ID}, corresponding to CAN IDs. The number of subsets from a CAN traffic trace becomes equal to the number of distinct CAN IDs in the CAN traffic trace. Unlike READ, our method leverages the total bit-flip rates just to find a location where additional analysis can be performed. As described in Section III, the total bit-flip rates are insufficient in identifing signal boundaries.

For a particular subset in our method, the i-th data payload is denoted by M_i. Bit flips at each bit location are obtained from two consecutive CAN messages. If the k-th bit is zero (one) in M_{i-1} and the k-th bit is one (zero) in M_i, the k-th bit flip increases by one. A set of bit flips on each bit location which is computed from M_i and M_{i-1} is denoted by BF_i. The k_i of BF_i indicates whether or not the k-th bit of M_i is flipped from M_{i-1}. BF_i is obtained as follows. For $i \geq 1$, $M_i \in S_{ID}$,

$$BF_i = M_{i-1} \oplus M_i, \tag{1}$$

where $M_0 = 0$ and $1 \leq i \leq |S_{ID}|$. It should be noted that every CAN message has a common length (i.e., DLC) within a subset for correct bit-wise operation. As a result, the total number of bit flips on the k-th bit, $TotalBF_k$, and the total bit-flip rates on the k-th bit, $TotalBFR_k$, are obtained as follows.

$$TotalBF_k = \sum_{i=1}^{|S_{ID}|} k_i \tag{2}$$

$$TotalBFR_k = \frac{TotalBF_k}{|S_{ID}|}, \tag{3}$$

where $k_i \in \{0, 1\}$ is i-th bit value of BF_i. The pseudo-code for the first step of our method is given in Algorithm 1.

Algorithm 1: Total Bit-Flip Rate.

```
procedure Total Bit-flip Rate T: CAN traffic
    S_ID ← A subset of T for a particular CAN ID
    for i ← 1 to |S_ID| do
        M_i ∈ S_ID ← i-th data payload
        BF_i ← M_{i-1} ⊕ M_i
        /* k_i ∈ {0,1} k-th bit in BF_i */
        TotalBF_k ← TotalBF_k + k_i
    TotalBFR ← TotalBF / |S_ID|
    return TotalBFR
```

B. Checksum / Counter Signal Identification

In a CAN data payload, checksum and counter both have clear patterns by which signal boundaries are easily identified. Since checksum is a signal which is used to detect a transmission error, it should be always the same value for the same data payload. Our method leverages a key-value data structure (a.k.a. a map, a dictionary, or a hash) to see if checksum is used in a data payload.

Counter is used to synchronize transmissions between a transmitter and a receiver (or multiple receivers). Commonly, the counter signal always increases by one. Accordingly, it is easy to identify the counter signal. Once the signal does not increase by one, our method rejects the boundary for counter signal. In addition, our method finds a bit location where the bit-flip rate is one. In common, a bit-flip rate of one means the least significant bit (LSB) of counter. The pseudo-code for checksum and counter signal boundaries is given in Algorithm 2.

C. Physical Value Signal Boundary Analysis

Unlike the existing methods that roughly determine physical value signal boundaries [17], [18], our method conducts additional analysis to identify precise signal boundaries. Our method leverages the time series of bit-flip rates as well as the total bit-flip rates. Because the bit-flip rates are affected

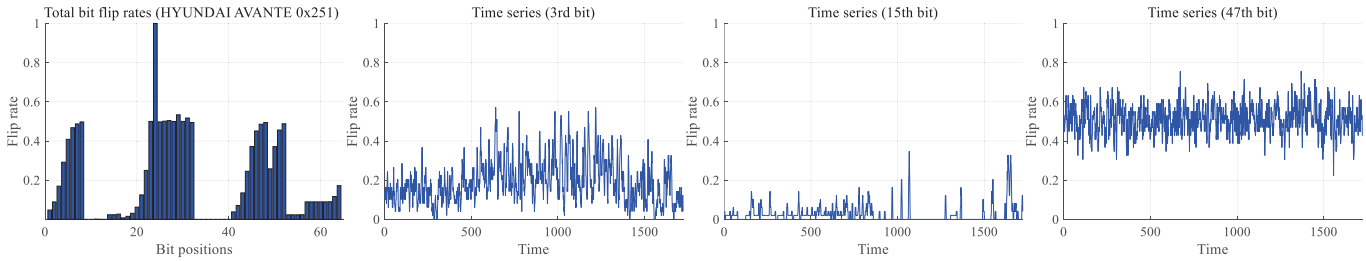


Fig. 5. Total bit-flip rates and its time series.

Algorithm 2: Checksum / Counter Signal Boundaries.

procedure ChecksumS_{ID}: A subset of a CAN traffic, $TotalBFR$: Total bit-flip rates
  $csFrom \leftarrow$ find$(TotalBFR > thres_{cs})$
  $csTo \leftarrow csFrom$
  while $(True)$**do**
    for all $M_i \in S_{ID}$**do**
      /* Extraction of key and value */
      $key \leftarrow$ GetKey$(M_i, csTo, csFrom)$
      $val \leftarrow$ GetVal$(M_i, csTo, csFrom)$
      if $hashTable[key] == null$**then**
        $hashTable[key] \leftarrow val$
      else if $hashTable[key]\ != val$**then**
        if $csTo < csFrom$
          /* Boundary of Checksum Signal */
          $csTo \leftarrow csTo + 1$
          return $[csTo\ csFrom]$
        else
          return $null$ /* No Checksum */
    $csTo \leftarrow csTo - 1$

procedure Counter$TotalBFR$: Total bit-flip rates
  $ctFrom \leftarrow$ find$(TotalBFR == 1)$
  if $ctFrom == null$
    return $null$ /* No Count */
  $ctTo \leftarrow ctFrom$
  while $(True)$
    if $TotalBFR_{ctTo-1} * 2 == TotalBFR_{ctTo}$**then**
      $ctTo \leftarrow ctTo - 1$
    else
      /* Boundary of Counter Signal */
      return $[ctTo\ ctFrom]$

Algorithm 3: Physical Value Signal Boundaries.

procedure PHYSICAL_VALUES_{ID}: A subset of a CAN traffic flow, $[csTo\ csFrom]$: The checksum signal range, $[ctTo\ ctFrom]$: The counter signal range
  $setOfSig \leftarrow$ list()
  $setOfSig.add([csTo\ csFrom], [ctTo\ ctFrom])$
  $< BFR_1, ..., BFR_{DLC} > \leftarrow$ GetTimeSeries(S_{ID})
  for $i \leftarrow 1$ to $DLC - 1$ **do**
    $corr(i) \leftarrow$ GetCorrelations(BFR_i, BFR_{i+1})
  $sigTo \leftarrow 1$
  $sigFrom \leftarrow sigTo$
  while $sigFrom < DLC$**do**
    if $sigFrom + 1 == csTo$ **then**
      /* Physical Signal Boundary */
      $setOfSig.add([sigTo\ sigFrom])$
      $sigTo \leftarrow csFrom + 1$
      $sigFrom \leftarrow sigTo$
    if $sigFrom + 1 == ctTo$ **then**
      /* Physical Signal Boundary */
      $setOfSig.add([sigTo\ sigFrom])$
      $sigTo \leftarrow ctFrom + 1$
      $sigFrom \leftarrow sigTo$
    if $corr(sigFrom) < thres_{corr}$ **then**
      /* Physical Signal Boundary */
      $setOfSig.add([sigTo\ sigFrom])$
      $sigTo \leftarrow sigFrom + 1$
      $sigFrom \leftarrow sigTo$
    else
      $sigFrom \leftarrow sigFrom + 1$
  return $setOfSig$

by physical movement of a vehicle, they are not constant. For example, when a vehicle accelerates, the corresponding physical value is affected by physical movement. Our core concept is to observe whether or not the bit-flip rates of two consecutive bits present a similar trend. Our method additionally separates a subset, S_{ID}, into multiple sub-subsets via a sliding window. We denote by BFR_k the bit-flip rate on the k-th bit location. Fig. 5 shows an example of the total bit-flip rates from S_{ID} and the time series of bit-flip rates on the k-th bit, which is obtained via a sliding window manner.

Our main concept for precise identification of signal boundaries is to analyze the correlation between the bit-flip rate time series of two consecutive bits. If two consecutive bits are assigned to the same physical value, it follows that the respective bit-flip rate time series would follow a similar trend. That is, when one bit-flip rate increases (or decreases), the other bit-flip rate also increases (or decreases). However, one condition is that the more significant bit sometimes does not change even though the less significant bit changes. For example, a particular signal repeats an addition by one and subtraction by one. When this is the case, the bit-flip rate of the least significant bit (LSB) is only high, whereas the most significant bit (MSB) would be constant. Our method only analyzes the correlation of a partial time series corresponding to when the bit-flip rate of the MSB changes. The pseudo-code for the last step of our method is given in Algorithm 3.

Fig. 6. Experimental Setup for CAN message measurement (left) and the three real vehicles. (a) CAN message measurement via OBD-II port. (b) 2017 KIA K5. (c) 2018 GENESIS EQ900. (d) 2020 HYUNDAI AVANTE.

VI. Evaluation

A. Dataset Description

As the reference for the evaluation of our method, we use OpenDBC [19] which is publicly available online. As mentioned before, it is difficult for researchers to obtain a real DBC format file kept in confidential by car manufacturers. OpenDBC can be one of the alternatives to this concern. Moreover, READ [17] was also evaluated with OpenDBC. We used the specification for Hyundai Kia generic. To obtain the corresponding dataset, we measured CAN messages from the three vehicles, Kia K5, Hyundai Avante, and Genesis EQ900 that are all from Hyundai and Kia motors group. We observed any possible long-term changes in these vehicles so that CAN messages were measured over 1 h.

B. Checksum Signal Identification

Our method identifies a checksum signal by using a hash table that checks whether or not checksum has been satisfied. Since checksum is usually used to detect transmission errors, it must always be the same value for the same payload. Our method searches for points where the bit-flip rate is more than a rate of 0.3 and extends its candidate boundary until the checksum condition is satisfied. When this condition can no longer be satisfied, our method rejects the checksum signal boundaries. While searching for an efficient checksum signal, we also discovered an interesting property. In our dataset, the checksum signals are usually located immediately following a counter signal or at the end of the payload. To save execution time, we added this additional condition when checksum is identified by our method. Table I shows the list of CAN IDs used in our evaluation and their checksum signal boundaries. Our method correctly identifies the same boundaries as those specified in OpenDBC.

In contrast, unlike our method, READ is designed to identify checksum signals based on total bit-flip rates. The reason why the READ method is not able to properly identify signals for checksum is that total bit-flip rates of checksum signals are always not distributed at a rate of 0.5. From this result, we conclude that our method is generally able to identify a checksum signal in CAN messages.

C. Counter Signal Identification

The signals for counter have a clear pattern from which a boundary can be easily identified. Since the counter signal increases by one, its LSB always changes from zero (one) to one (zero). Moreover, the difference between a previous value and the current one is always one. To identify a counter signal, our method searches for a location where the total bit-flip rate is one. Beginning at this location, our method extends the boundary until the condition for counter signal is satisfied. READ and our method both perfectly identify counter signals in our dataset. Table II shows the list of CAN IDs used in our experimental vehicle and the boundaries of the counter signals. Fig. 7 shows an example of total bit-flip rates for CAN IDs, 0x251 from Kia K5, 0x220 from Genesis EQ900, and 0x387 from Hyundai Avante. From this total bit-flip rate, it can be roughly seen that where signals for counter and checksum are located.

D. Physical Value Signal Identification

By analyzing time series of bit-flip rates at each bit location, our method identifies signal boundaries assigned to represent a physical value. Since the READ method is designed to identify a signal boundary based on total bit-flip rates, its accuracy would be affected by the status of a vehicle the moment a dataset is collected. Even if two different datasets originate from the same vehicle, the results differ according to the condition when they are collected. The READ method would have different results from the two different datasets if one was measured when the vehicle was much more idling than driving and the other one when the vehicle was much more in motion than idling. Accordingly, the functions used in a vehicle should evenly operate during the measurement in order for READ to correctly identify boundaries of signals. That is why, in their evaluations, much CAN traffic operates every function of a vehicle. We believe that trends in bit-flip rates between two consecutive bits follow the same trend if they are assigned to a signal representing the same physical value.

Fig. 8 shows the time series of bit-flip rates on a bit position (left) and the correlation of two consecutive bits (right). As can be seen in the figure, bit-flip rates change as a function of time because the vehicles were in motion and idling when the dataset was collected, which implies that the corresponding functions were initiated. It should be noted that our method does not need to equally operate every function in the vehicle when the dataset is collected. In the figure, it can be seen that the bit-flip rate for a lower bit only increases, while the bit-flip rate for a higher bit does not. Conversely, it is rare for the bit-flip rate of a higher bit to increase whereas those of a lower bit do not increase if the two bits are assigned to represent the same physical value. Though this is a natural property, it actually reduces the degree of correlation between bit-flip rates, contrary to our assumption that the bit-flip rates would increase and

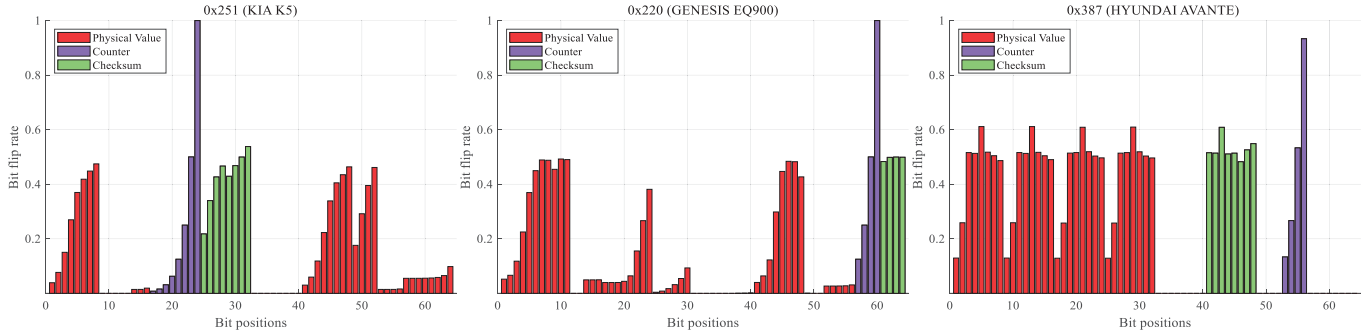


Fig. 7. Total bit-flip rates at each bit location.

TABLE I
RESULT OF IDENTIFIED CHECKSUM SIGNALS

CAN ID	Where CAN ID is used	OpenDBC [19]	READ [17]	Our Method
0x111	KIA K5	57-60	-	57-60
0x112	KIA K5	9-10	-	9-10
0x130	GENESIS EQ900	57-64	2-8, 57-64	57-64
	HYUNDAI AVANTE		1-6, 57-64	
0x140	GENESIS EQ900	57-64	1-8, 57-64	57-64
	HYUNDAI AVANTE		1-6, 57-64	
0x220	KIA K5	61-64	2-11, 61-64	64-64
	GENESIS EQ900		3-44, 61-64	
	HYUNDAI AVANTE		2-11, 61-64	
0x251	KIA K5	25-32	25-32	25-32
	GENESIS EQ900			
	HYUNDAI AVANTE			
0x260	KIA K5	61-64	61-64	61-64
	GENESIS EQ900			
	HYUNDAI AVANTE			
0x2B0 (*)	KIA K5	33-36	-	33-36
	GENESIS EQ900			
	HYUNDAI AVANTE		41-48	41-48
0x381	KIA K5	49-56	49-56	49-56
	GENESIS EQ900		53-56	
	HYUNDAI AVANTE		49-56	
0x387 (*)	KIA K5	41-48	-	41-48
	GENESIS EQ900		53-56	
	HYUNDAI AVANTE		1-32, 41-48, 53-56	
0x394	GENESIS EQ900	53-56	2-8, 53-55	53-56
	HYUNDAI AVANTE		53-56	
0x484	GENESIS EQ900	-	-	-
	HYUNDAI AVANTE			
0x4F1	GENESIS EQ900	-	2-3	-
	HYUNDAI AVANTE			
0x556	KIA K5	-	-	-
0x557	KIA K5	-	-	-
0x5A0	KIA K5	-	41-44, 51-52, 54-56	-

*indicates the case that we make sure that OpenDBC is incorrect reference because information for DLC is different from ones from the vehicles.

TABLE II
RESULT OF IDENTIFIED COUNTER SIGNALS

CAN ID	Where CAN ID is used	OpenDBC [19]	READ [17]	Our Method
0x111	KIA K5	61-62	61-62	61-62
0x112	KIA K5	11-12	11-12	11-12
0x130	GENESIS EQ900	53-56	53-56	53-56
	HYUNDAI AVANTE			
0x140	GENESIS EQ900	53-56	53-56	53-56
	HYUNDAI AVANTE			
0x220	KIA K5	57-60	57-60	57-60
	GENESIS EQ900			
	HYUNDAI AVANTE			
0x251	KIA K5	17-24	17-24	17-24
	GENESIS EQ900			
	HYUNDAI AVANTE			
0x260	KIA K5	59-60	59-60	59-60
	GENESIS EQ900			
	HYUNDAI AVANTE			
0x2B0 (*)	KIA K5	37-40	37-40	37-40
	GENESIS EQ900			
	HYUNDAI AVANTE			
0x381	KIA K5	41-48	41-48	41-48
	GENESIS EQ900			
	HYUNDAI AVANTE			
0x387 (*)	KIA K5	-	-	-
	GENESIS EQ900		53-56	53-56
	HYUNDAI AVANTE			
0x394	GENESIS EQ900	9-11	9-11	9-11
	HYUNDAI AVANTE			
0x484	GENESIS EQ900	-	-	-
	HYUNDAI AVANTE		3-6	3-6
0x4F1	GENESIS EQ900	25-28	25-28	25-28
	HYUNDAI AVANTE			
0x556	KIA K5	-	-	-
0x557	KIA K5	-	-	-
0x5A0	KIA K5	-	-	-

*indicates the case that we make sure that OpenDBC is incorrect reference because information for DLC is different from ones from the vehicles.

decrease in tandem. For this reason, we only compute correlation on sets where the bit-flip rate for a higher bit increases, which is indicated in red in the figure. From this result, our method is able to identify a boundary between two different signals. For example, our method identifies a signal boundary based on the correlation between two consecutive bits. For the 48th and 49th bits from CAN ID 0x251 of the Kia K5 2017, the time series of bit-flip rates is zero, and accordingly, our method determines the 48th bit to be the end of a signal and the 49th bit as the start of another signal. In addition, Fig. 9 shows an example

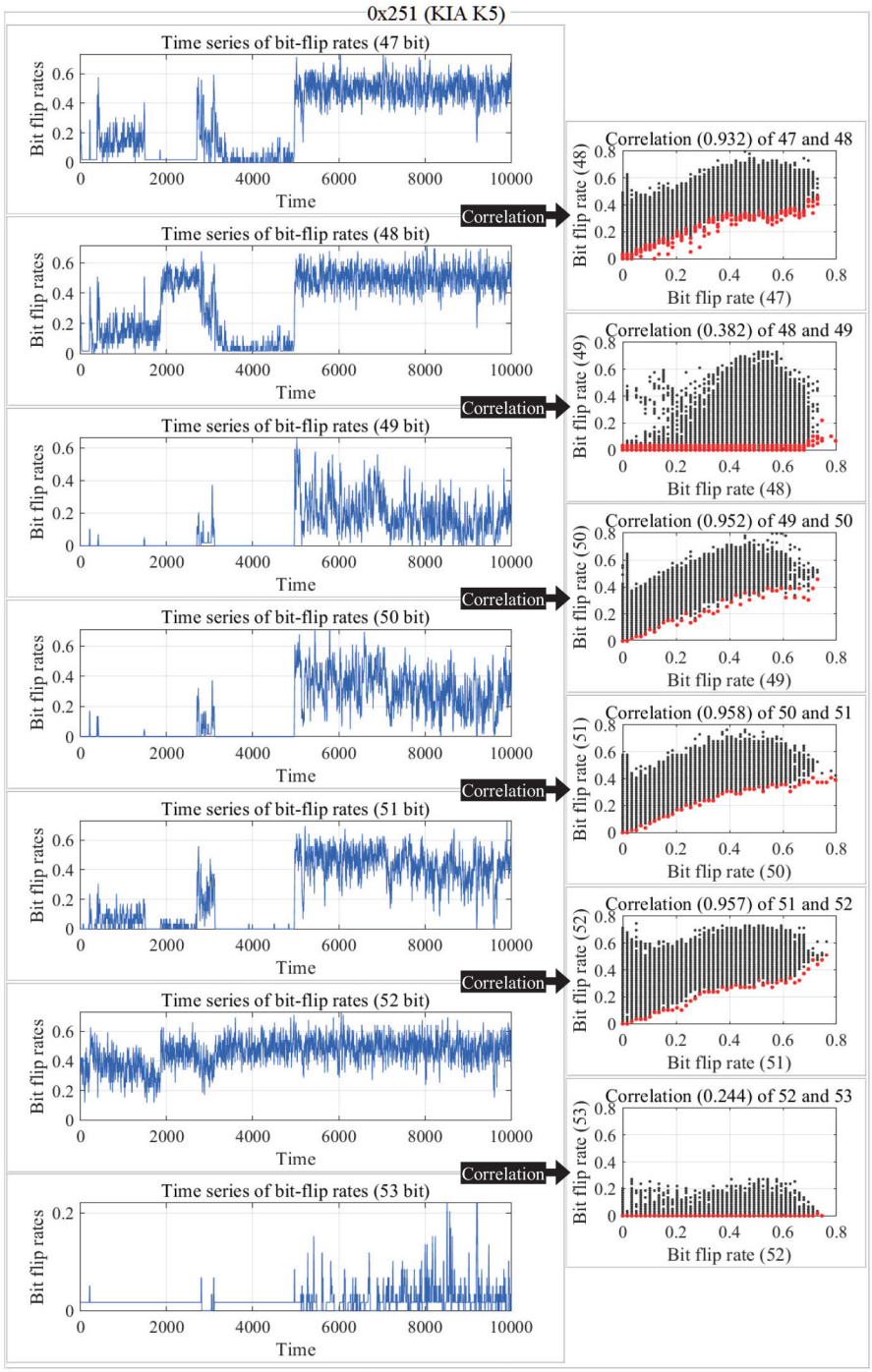


Fig. 8. Correlations of bit-flip rate time series from consecutive two bit positions.

of visualization for the comparison of the READ method and our method with a reference of OpenDBC. It is clear that our method identifies more signal boundaries than READ, as our method sets a boundary based on the correlation even though the total bit-flip rates of two consecutive bits are similar. We implemented the READ algorithm to compare our method with it. READ and our method both do not produce the same result as OpenDBC, most likely due to the fact that we used the generic DBC format file for Hyundai and Kia vehicles, but collected dataset from specific vehicles, Kia K5 2017, Genesis EQ900 2018, and Hyundai Avante 2020 even though they are all from the Hyundai and Kia motors group. It seems that not all the functions are included in the vehicle used for this study. In addition, READ does not identify any constant or unused fields even though these fields can be defined in DBC format files. Table III shows the number of identified signals by the READ method and our method. It can be seen that our method is able to identified more number of signals than the READ method. As a result, we conclude that our method outperforms existing techniques for signal boundary identification.

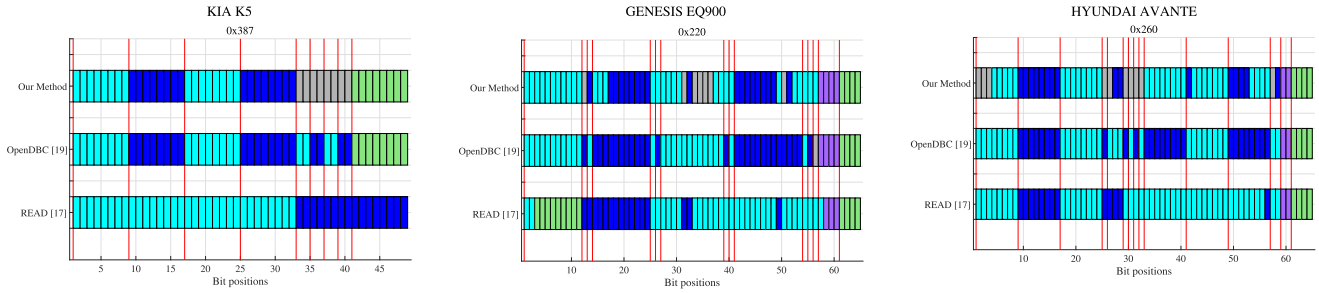


Fig. 9. Signal boundaries identified by our method.

TABLE III
NUMBER OF ALL IDENTIFIED SIGNALS

CAN ID	Where CAN ID is used	OpenDBC [19]	READ [17]	Our Method
0x111	KIA K5	13	9	10
0x112	KIA K5	12	9	11
0x130	GENESIS EQ900	7	9	9
	HYUNDAI AVANTE		7	10
0x140	GENESIS EQ900	8	8	10
	HYUNDAI AVANTE		7	10
0x220	KIA K5	14	12	17
	GENESIS EQ900		10	17
	HYUNDAI AVANTE		10	17
0x251	KIA K5	11	6	10
	GENESIS EQ900		10	16
	HYUNDAI AVANTE		8	13
0x260	KIA K5	15	12	16
	GENESIS EQ900		9	14
	HYUNDAI AVANTE		9	16
0x2B0 (*)	KIA K5	5	5	6
	GENESIS EQ900		6	5
	HYUNDAI AVANTE		6	8
0x381	KIA K5	14	7	10
	GENESIS EQ900		10	15
	HYUNDAI AVANTE		8	12
0x387 (*)	KIA K5	9	2	6
	GENESIS EQ900		8	6
	HYUNDAI AVANTE		6	6
0x394	GENESIS EQ900	25	17	21
	HYUNDAI AVANTE		12	18
0x484	GENESIS EQ900	10	8	16
	HYUNDAI AVANTE		11	16
0x4F1	GENESIS EQ900	12	8	12
	HYUNDAI AVANTE		9	14
0x556	KIA K5	6	8	10
0x557	KIA K5	6	4	6
0x5A0	KIA K5	14	13	16

*indicates the case that we make sure that OpenDBC is incorrect reference because information for DLC is different from ones from the vehicles.

VII. DISCUSSION

Limited Information for DBC Format Files. As mentioned at the outset of this paper, car manufacturers are keeping DBC format files confidential for real vehicles. Accordingly, we did not verify the results of our method. Even though the specifications in OpenDBC were used as a reference in our evaluation, the vehicle we used did not perfectly match the DBC format file from OpenDBC. We expect that our method would show better performance with access to precise information.

Multiple types of vehicles. We have evaluated our method on a dataset collected from a single vehicle. Commonly, it is accepted that DBC format files for two different vehicles models and types produced by the same manufacturer are compatible. Some unused or constant fields may be assigned for certain functions in other models by the same car manufacturer. We expect that our method would show better performance with datasets collected from multiple types of vehicles.

Undocumented Signal. Even if the specifications in OpenDBC exactly match the ones used by vehicle manufacturers, there is a possibly of an undocumented signal. Since DBC format files are usually created by several departments within one manufacturer, a signal can be used without proper documentation in the DBC specification. Because of this concern, it is difficult to properly evaluate the methods for signal boundary identification, including our proposed method. However, it is expected that undocumented signals are much more likely to be identified by our method.

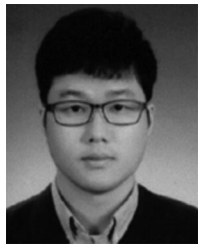
VIII. CONCLUSION

In this paper, we have presented a method that is able to identify signal boundaries assigned in a data payload. Unlike state-of-the-art READ, which uses total bit-flip rates, our method analyzes the time series of bit-flip rates in each bit position. Based on the correlation of time series between two consecutive bits, signal boundaries are determined. Recently, a number of electronic components such as the Internet and cellular network are being readily connected with vehicles. For this reason, studies in automotive security are increasingly important academia. However, most of these studies have not been able to fully produce comprehensive results because car manufacturers maintain a hold over information on the DBC format files. In light of this, we expect our method to greatly assist future research in automotive security even without the aid of the DBC format files.

REFERENCES

[1] "Keen security lab of tencent," Accessed: Apr. 30, 2019. [Online]. Available: https://keenlab.tencent.com/en/2016/09/19/Keen-Security-Lab-of-Tencent-Car-Hacking-Research-Remote-Attack-to-Tesla-Cars/
[2] C. Miller and C. Valasek, "Remote exploitation of an unaltered passenger vehicle," *Black Hat USA*, vol. 2015, p. 91, 2015.
[3] C. Miller and C. Valasek, "Adventures in automotive networks and control units," *Def. Con.*, vol. 21, pp. 260–264, 2013.
[4] S. Checkoway *et al.*, "Comprehensive experimental analyses of automotive attack surfaces," in *Proc. USENIX Secur. Symp.*, 2011, pp. 447–462.
[5] K. Koscher *et al.*, "Experimental security analysis of a modern automobile," in *Proc. IEEE Symp. Secur. Privacy.*, 2010, pp. 447–462.

[6] "Serial communication—Hack your canbus," Accessed: Aug. 10, 2018. [Online]. Available: https://www.youtube.com/watch?v=1kd5nl0Akfo
[7] T. Hoppe, S. Kiltz, and J. Dittmann, "Security threats to automotive can networks-practical examples and selected short-term countermeasures," in *Proc. Int. Conf. Comput. Saf., Rel., Secur*. Berlin, Germany: Springer, 2008, pp. 235–248.
[8] C. Ling and D. Feng, "An algorithm for detection of malicious messages on can buses," in *Proc. Nat. Conf. Inf. Technol. Comput. Sci*., 2012.
[9] A. Taylor, N. Japkowicz, and S. Leblanc, "Frequency-based anomaly detection for the automotive can bus," in *Proc. World Congr. Ind. Control Syst. Secur*., 2015, pp. 45–49.
[10] M. R. Moore, R. A. Bridges, F. L. Combs, M. S. Starr, and S. J. Prowell, "Modeling inter-signal arrival times for accurate detection of can bus signal injection attacks: A data-driven approach to in-vehicle intrusion detection," in *Proc. 12th Annu. Conf. Cyber Inf. Secur. Res*., 2017, pp.. 1–4.
[11] K.-T. Cho and K. G. Shin, "Fingerprinting electronic control units for vehicle intrusion detection," in *Proc. 25th {USENIX} Secur. Symp.*, 2016, pp. 911–927.
[12] W. Choi, H. J. Jo, S. Woo, J. Y. Chun, J. Park, and D. H. Lee, "Identifying ecus using inimitable characteristics of signals in controller area networks," *IEEE Trans. Veh. Technol.*, vol. 67, no. 6, pp. 4757–4770, Jun. 2018.
[13] W. Choi, K. Joo, H. J. Jo, M. C. Park, and D. H. Lee, "Voltageids: Low-level communication characteristics for automotive intrusion detection system," *IEEE Trans. Inf. Forensics Security*, vol. 13, no. 8, pp. 2114–2129, Aug. 2018.
[14] P.-S. Murvay and B. Groza, "Source identification using signal characteristics in controller area networks," *IEEE Signal Process. Lett.*, vol. 21, no. 4, pp. 395–399, Apr. 2014.
[15] H. Lee, K. Choi, K. Chung, J. Kim, and K. Yim, "Fuzzing can packets into automobiles," in *Proc. IEEE 29th Int. Conf. Adv. Inf. Netw. Appl*., 2015, pp. 817–821.
[16] S. Bayer and A. Ptok, "Don't fuss about fuzzing: Fuzzing controllers in vehicular networks," *13th Escar Europe*, p. 88.
[17] M. Marchetti and D. Stabili, "Read: Reverse engineering of automotive data frames," *IEEE Trans. Inf. Forensics Secur.*, vol. 14, no. 4, pp. 1083–1097, Apr. 2019.
[18] M. Markovitz and A. Wool, "Field classification, modeling and anomaly detection in unknown can bus networks," *Veh. Commun.*, vol. 9, pp. 43–52, 2017.
[19] "Opendbc from comma.ai," Accessed: Sep. 7, 2019. [Online]. Available: https://github.com/commaai/opendbc
[20] K.-T. Cho and K. G. Shin, "Error handling of in-vehicle networks makes them vulnerable," in *Proc. ACM SIGSAC Conf. Comput. Commun. Secur*., 2016, pp. 1044–1055.
[21] A.-I. Radu and F. D. Garcia, "Leia: A lightweight authentication protocol for can," in *Proc. Eur. Symp. Res. Comput. Secur*., Berlin, Germany: Springer, 2016, pp. 283–300.
[22] B. Groza, S. Murvay, A. Van Herrewege, and I. Verbauwhede, "Libracan: A lightweight broadcast authentication protocol for controller area networks," in *Proc. Int. Conf. Cryptology Netw. Secur*., Berlin, Germany: Springer, 2012, pp. 185–200.
[23] A. Greenberg, "Hackers remotely kill a jeep on the highway-with me in it," *Wired*, vol. 7, p. 21, 2015.
[24] S. U. Sagong, X. Ying, A. Clark, L. Bushnell, and R. Poovendran, "Cloaking the clock: Emulating clock skew in controller area networks," in *Proc. 9th ACM/IEEE Int. Conf. Cyber-Phys. Syst*., Piscataway, NJ, USA: IEEE Press, 2018, pp. 32–42.
[25] M. Kneib and C. Huth, "Scission: Signal characteristic-based sender identification and intrusion detection in automotive networks," in *Proc. ACM SIGSAC Conf. Comput. Commun. Secur*., 2018, pp. 787–800.
[26] D. K. Nilsson and U. E. Larson, "Conducting forensic investigations of cyber attacks on automobile in-vehicle networks," *Int. J. Digit. Crime Forensics*, vol. 1, no. 2, pp. 28–41, 2009.
[27] H. Mansor, K. Markantonakis, R. N. Akram, K. Mayes, and I. Gurulian, "Log your car: The non-invasive vehicle forensics," in *Proc. IEEE Trustcom/BigDataSE/ISPA.*, 2016, pp. 974–982.
[28] C. Smith, *The Car Hacker's Handbook: A Guide for the Penetration Tester*. No Starch Press, 2016.
[29] H. M. Song, H. R. Kim, and H. K. Kim, "Intrusion detection system based on the analysis of time intervals of can messages for in-vehicle network," in *Proc. Int. Conf. Inf. Netw*., 2016, pp. 63–68.
[30] "Vector, dbc file format documentation," 2007.



Wonsuk Choi received the B.S. degree in mathematics from the University of Seoul, Seoul, South Korea, in 2008, and the M.S. and Ph.D. degrees in information security from Korea University, Seoul, South Korea, in 2013 and 2018, respectively. He was a Postdoctoral Researcher with the Graduate School of Information Security, Korea University. In 2020, he joined the Division of IT Convergence Engineering with Hansung University, Seoul, South Korea, as an Assistant Professor. His research interests include security for body area networks, usable security, applied cryptography, and smart car security.

Seyoung Lee received the B.S. degree in mathematics (major) and computer science (bimajor) from the University of Seoul, Seoul, South Korea, in 2014, and the M.S. degree in information security in 2016, from the Graduate School of Information Security of Korea University, Seoul, South Korea, where he is currently working toward the Ph.D. degree in information security. His research interests include applied cryptography, automotive security, and senor security.

Kyungho Joo received the B.S. degree from the College of Information and Communication in 2016, and the M.S. degree in information security in 2018 from Korea University, Seoul, South Korea, where he is currently working toward the Ph.D. degree in information security with the Graduate School of Information Security. His research interests include Vehicular-IT security and Wireless security.

Hyo Jin Jo received the B.S. degree in industrial engineering and the Ph.D. degree in information security from Korea University, Seoul, South Korea, in 2009 and 2016, respectively. From 2016 to 2018, he was a Postdoctoral Researcher with the Department of Computer and Information Systems, the University of Pennsylvania, Philadelphia, PA, USA. He is currently an Assistant Professor with the School of Software, Soongsil University, Seoul, South Korea. His research interests include applied cryptography, security and privacy for ad-hoc networks, and IoT or CPS security.

Dong Hoon Lee (Member, IEEE) received the B.S. degree from the Department of Economics, Korea University, Seoul, South Korea, in 1985, and the M.S. and Ph.D. degrees in computer science from the University of Oklahoma, Norman, OK, USA, in 1988 and 1992, respectively. He is currently a Professor and the Director with the Graduate School of Information Security, Korea University. Since 1993, he has been with the Faculty of Computer Science and Information Security, Korea University. His research interests include cryptographic protocol, applied cryptography, functional encryption, software protection, mobile security, vehicle security, and ubiquitous sensor network (USN) security.